Chapter 5

Social-influence modeling in Python using defSim

This chapter is based on joint work with Anton Laukemper and Dieko Bakker.

Model alignment and code sharing are essential next steps to advance our understanding of agent-based models, and build common ground within strands of literature. The rich literature on social-influence models has brought forth valuable insights into the dynamics of community belief formation and opinion exchange. Yet, the vast number of slight variations in model assumptions and conceptualizations make it hard to readily compare outcomes across models.

Here, we argue why and how a general simulation framework could help to achieve the goal of harmonizing the agent-based modeling literature, and introduce one such example of modeling software to inspire others. We present a framework for discrete-event social-influence modeling in Python called defSim. This framework is designed with a focus on modularity, extendability, scalability, and simplicity. As such, it is a multi-purpose tool, well-suited for extensive simulation experiments, model comparison, replication, and teaching. This chapter elaborates defSim's principles, software architectural foundations, and current functionality, and demonstrates how model comparison with defSim can be achieved by studying different communication regimes in metric and nominal opinion models. Specifically, we study the emergence of isolation and fragmentation in models with one-to-many communication, a communication regime omnipresent in online social media. Finally, we provide research directions that follow naturally from the creation of this framework and call on our colleagues to exploit the possibilities of, criticize, comment on, and contribute to the modeling framework.

5.1 Introduction

Understanding opinion dynamics and the mechanisms behind opinion formation has been a major challenge in a multitude of scientific domains such as political science, sociology, psychology, network science, mathematics and computer science for many years (for reviews, see e.g. Flache et al., 2017; Castellano, Fortunato, & Loreto, 2009; Iyengar et al., 2019). In fear of increasingly polarized political climates within socially segregated societies, the yearly number of publications related to opinion dynamics has risen steeply in the past 25 years (see Figure 5.1). Studying opinions in social systems is notoriously difficult due to the interdependencies between individuals, as well as their embeddedness in society at large where politicians, pundits and collective events all factor into the observed opinion distributions (Macy & Willer, 2002). We may study individual reasoning, ponder argument-topic graphs or even examine large-scale longitudinal survey data, but rarely do we fully grasp the causal mechanisms that bring about polarization, ideological fragmentation or consensus (Keijzer & Mäs, in press). Agent-based models (ABMs) are a tool well suited to do exactly this. They allow the researcher to examine the internal validity of a theoretical argument, when the proposed process is embedded in a system that exhibits properties of a complex system (Edmonds et al., 2019). In other words, we can use ABMs as a means to understand macro-outcomes that result from many stochastic micro-level interactions constrained by and possibly interacting with a meso-level interaction structure (Macy & Flache, 2009).

In this chapter, we introduce *defSim*—the discrete event framework for social-influence models (Laukemper et al., 2019). defSim is a fully open source Python package that allows researchers, teachers and students to efficiently code and run social-influence models. defSim is mainly a simulation framework, but also contains a repository of pre-programmed, classic social-influence models. The software package can be useful to anyone who wants to model collective dynamics of social-influence processes on opinions, attitudes or beliefs and wishes to (1) sketch a quick idea in a few lines of code, (2) scale-up their code to run quickly in parallel on their own computer or a High-Performance Computing (HPC) cluster, (3) seek a simulation tool with easy to read syntax to introduce others (e.g.,students) to simulation-based research.

This simulation framework answers the call from a group of prominent modelers who identified the theoretical integration of social-influence models as one of the two main frontiers in opinion dynamics research (Flache et al., 2017). To create common ground "[...] modelers need to identify the critical assumptions and predictions of their models, and need to compare these assumptions as well as their formal implementation to existing models." (Flache et al., 2017, § 3.3). To advance our

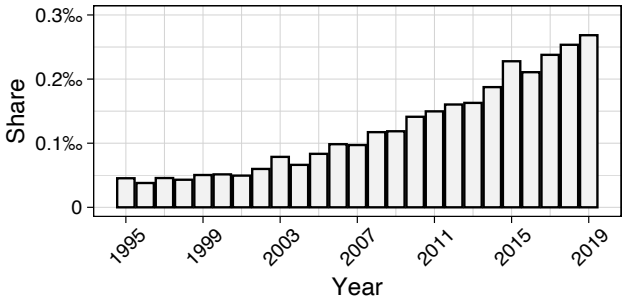


Figure 5.1 Share of publications on opinion dynamics by year (1995-2019). Retrieved through a topic search on "opinion dynamics" in the Web of Science database on May 4th, 2020, and normalized over the total number of publications listed in Web of Science per year.

understanding of the interplay of theoretical mechanisms within different modeling paradigms, it is critical to align models (Axtell et al., 1996). The software presented here is designed with this goal in mind. We identified seven basic principles of social-influence models and used modular programming techniques to reflect those principles in the software design.

We demonstrate the framework's capacity to compare models, by studying the generalization of a mechanism relevant for models of opinion dynamics in online social media. Online social media play a significant role in the exchange of ideas and opinions nowadays, and some have warned for its ability to polarize and segregate groups of different ideological basis (Pariser, 2011). Chapter 3 proposed an explanation for increased isolation and fragmentation online that rests purely on a difference in how communication is structured. Online communication is characterized by one-to-many communication—users sending messages (e.g.,posts, stories or tweets) to all of their contacts at once. This seemingly small difference of communication regime, has profound impact on system dynamics. Yet, to the best of our knowledge, communication regimes are understudied in the ABM literature, where the overwhelming majority of discrete-event models use one-to-one communication. Here, we study to what extent the mechanism from Chapter 3 generalizes, by comparing the effects of one-to-many and one-to-one communication across two prominent models of opinion dynamics: the (nominal) model for the dissemination of culture (Axelrod, 1997) and the (metric) bounded confidence model (Hegselmann & Krause, 2002; Deffuant et al., 2000).

The chapter is structured as follows. First, we elaborate on the need for a joint simulation framework (in Section 5.2), and explain the principles, structure and architecture of defSim (Section 5.3). Then, we demonstrate the desirability of a general modeling framework through a comparison of one-to-many influence effects

in nominal and metric opinion models (Section 5.4). We conclude with a discussion of the results and of our perspectives on advancing defSim—and social-influence models in general—in the future (Section 5.5).

5.2 Background

5.2.1 Versatility vs. usability

When it comes to coding an ABM, finding the optimum between versatility and usability of software applications is somewhat of a balancing act. As model code becomes more complex, it becomes increasingly difficult to understand what exactly is going on in the code, and how all its moving parts are interacting. There are excellent high-level programming languages or applications that make the creation of complex models easy through simple graphical interfaces, like, for example, NetLogo (Wilensky, 1999). Such software is very well suited for teaching purposes, but is outperformed by lower level programming languages in terms of computing power, ease of collaborating with others through, for example, distributed version control software, and code sharing in general. On the other end are low-level programming languages that may be very versatile, but require much more effort from secondary users to understand, check, and expand code. Likewise, the degree of specificity of a coding language or software package for particular research problems or modeling domains creates a similar trade-off. The smaller the domain, the easier it is to exploit the benefits of using predefined software functionality tailored specifically for models in that domain. Again, this usability for a particular application comes at the cost of versatility of the software for other domains.

Modular programming offers a flexible solution to the problem of finding the right optimum between versatility and usability for a given project. Simulation models can often be broken down into sequences of (repeating) events. Those events can then be transformed into modules that make up the building blocks of a simulation model. If the objects passed between the modules are the same regardless of the realization of the module, then modules can be freely replaced to create unique simulation models. Models that use pre-programmed modules are very easy to run and understand, making them well suited for teaching about social-influence models to people without much programming experience. However, projects that require a novel implementation of one model or a complete reordering of the presumed sequence can use the modules included in the modular package to keep their code as simple as possible, and hence understandable for others. For the desired level of flexibility, there is a matching level of modularity.

5.2.2 Model alignment

Whatever a researcher's modeling purpose and modeling strategy may be, building ABMs that offer reasonable representations of society while remaining informative and intelligible is challenging (Edmonds et al., 2019). On the one hand, we want to mimic the phenomenon of interest as best as we can; underlining that the studied mechanisms are responsible for the state of the system we wish to explain. On the other hand, complex models with many parameters and mechanisms are not easily interpretable, and may run the risk of not rigorously establishing causality within the proposed model.

A proper strategy for building a rich model of any social process, that does remain comprehensible, is by extending and adding to already existing models. Researchers may employ the TAPAS principle: to Take A Previous model and Add Something (Frenken, 2006). This relieves the researcher of the burden to understand all model elements, allowing to focus most effort on the novel extension. Furthermore, the baseline model should replicate the original finding, integrating replication into the standard workflow of innovative research. Though it may not be the most exposed part of scientific research, replication is fundamental to the model's internal validation and verification of model dynamics (Squazzoni, 2012). Rigorous extensions of the literature are obtained by complete *alignment* of models (Axtell et al., 1996). Axtell (pp. 124) convincingly puts forward that "[a]lignment is essential to support two hallmarks of cumulative disciplinary research: critical experiment and subsumption. If we cannot determine whether or not two models produce equivalent results in equivalent conditions, we cannot reject one model in favor of another that fits data better; nor are we able to say that one model is a special case of another more general one—as we do when saying Einstein's treatment of gravity subsumes Newton's."

The social-influence literature has embarked on this journey. Basic social-influence models have been developed by some, and others have extended those models with a variety of different concepts. Yet, to satisfy Axtell's criterion of alignment, an extension should not merely be loosely based on a foundational model. Rather, all elements that are not essential to the extension should match the foundational model *exactly*. Here lies a major challenge for the agent-based modeling community in general, and the opinion dynamics modelers in particular. Currently, few publications make their code publicly available. A 2017 review of 2367 articles indexed in Web of Science, retrieved with the search term "agent-based model*", showed that only 10% of these publications had made their code publicly available (Janssen, 2017). Generally, publications that do not provide the source code will include a representation of the model in another form, for example as a text description,

formal description, or pseudo-code. However, translating such a *conceptual model* into a concrete *implemented model* is not as straightforward as it may seem (Rand & Wilensky, 2006). In the absence of source code, the researcher who wishes to replicate an established model may run into problems when small errors creep into the code (see e.g., Van de Rijt et al., 2009, who discovered an error in a highly visible publication) or when the informal description of what the model should do does not match the formal implementation (see e.g.,the failed replication and discussion of another highly visible publication by Will & Hegselmann, 2008; Will, 2009). Even when the researcher is not explicitly mistaken "[c]omputer programs may imply assumptions we did not want to implement [...]. Rigourous replication can detect such problems and furthermore uncover the assumptions that lead to published results." (Will, 2009).

Adopting a shared framework for simulation as a community can help prevent, detect, and clarify such episodes. Communicating model assumptions, code sharing, and avoiding small mistakes is much easier if one can refer to a piece of software freely available online. Versioning and issue discussions, furthermore, help to identify and debate critical choices in the design process and their motivations.

5.3 A social-influence modeling framework for Python

defSim is a Python-based software package that presents the user with a discrete event framework for social-influence models. In this section we lay out the most important principles on which defSim is built (from Paragraph 5.3.1 onwards), address the core of the software architecture design choices (from Paragraph 5.3.2), and provide concise descriptions of the package contents, starting at Paragraph 5.3.3.

5.3.1 Principles

defSim accommodates agent-based modeling of social influence and is intended to be as flexible as possible in that domain. To make the wide array of implemented assumptions mutually compatible, we have to follow a few basic principles that all social-influence models implemented in defSim v0.1 should adhere to. It is very well possible that future versions of defSim will be able to extend these principles, increasing defSim's versatility. Furthermore, the modular use of defSim allows the user to exchange and extend the current principles on their own account as well.

Models are agent-based. Agents (or nodes) are the acting units in the models. We typically think of agents as humans, but this does not have to be true. ABMs have been employed in many different fields, from biology to computer science, and from

sociology to mathematics. Although defSim is tailored to social-influence models on beliefs and attitudes, many of its principles and insights will apply in other fields too.

Agents have features. Agents possess F features that may be subject to influence, govern receptiveness to influence, or both. These features can be continuous or categorical. Continuous features represent the metric view on opinions $o_{iF} \in [0, 1]$ (as used in e.g., Deffuant et al., 2000; Hegselmann & Krause, 2002; Lorenz, 2007). Categorical features may represent attitudes or general non-ordinal features with Q traits: $o_{iF} \in \{0, 1, ..., Q-1\}$ (as used in e.g., Axelrod, 1997). By convention, automatically generated features are labeled $f01$, $f02$, and so forth.

Agents live on a physical network. Agents are connected to each other through some pre-defined network. Agents are only able to interact when they are connected through a link (or edge). The shape of the network can be randomly generated based on some network generation principle (e.g.,one of the functions available in NetworkX), user defined through provision of an edgelist, or trivial by defining a complete graph. Furthermore, the network does not have to be static, but may change over time.

Agents live on an attribute distance network. Most influence functions will take the attribute distance between agents into account. attribute distance is a function of the overlap of a set of features between two agents who share a link in the physical network. Their attribute distance is thus a feature of the links in the physical network (denoted 'dist' in defSim). The implication of modeling their attribute distances this way, is that a attribute distance network emerges on top of the physical opportunity network. When both networks are utilized, this mimics the real world phenomenon that people might meet many others, but only a subset of these others are considered influential for shaping an individual's beliefs and attitudes.

Time is divided into discrete events. In each simulation run, there is an initialization phase and a simulation phase. In the simulation phase, communication occurs between agents as a series of communication events between (a) focal agent(s) and (a) receiving agent(s). The focal and receiving agents are typically chosen at random.

Communication is structured by a regime. The communication regime governs who is influencing whom at each time step. Most discrete time stochastic influence models have assumed one-to-one communication for its simplicity, but this is not a trivial assumption. Over the years, models have been developed that use many-to-one (Flache & Macy, 2011a) or one-to-many influence (Keijzer et al., 2018). Note that even though the agents involved in the exchange are labeled 'focal' and 'receiving', this does not mean that influence is necessarily uni-lateral because influence functions can include mutual influence.

Agents exert influence on each other. When agents are selected to interact at a time step, they exert influence on each other. The sending agent(s) will attempt to change a feature of the receiving agent(s). Typically, the success and/or strength of influence are determined by the attribute distance between agents, but this is ultimately up to the influence function picked or programmed by the user.

5.3.2 Software architecture

The framework is created in a way that combines multiple exchangeable components. Therefore, interfaces that prescribe input, output and behavior of each component need to be defined. Each component can then have multiple realizations that all implement the same interface. To achieve this we employ an interdependency design known as the factory method pattern (Gamma et al., 1995). This pattern helps in situations where a client (i.e. a method or class) expects a concrete implementation of an interface, but which implementation is used is variable. The factory method is implemented and made responsible for instantiating the implementations of the interface. The client can then request a specific instantiation from the factory method with the help of an identifier that determines which implementation is chosen. The factory method returns the concrete implementation according to the value of the identifier to the client, who can then use that object just like any other object. In defSim, the interfaces of the objects are defined via Python's abstract base classes (ABCs).

This pattern decouples the use of an object from its definition and creation. The client does not care how the object is created, and does not need to change if the object changes, as long as the object still satisfies the interface. If another implementation of the interface is developed, that implementation only needs to be added to the factory method. This simplifies maintenance of the code base.

Furthermore, we chose to create modules that are concerned with only a small, specialized part of the simulation run, following the interface-segregation principle (ISP) (Martin, 2003). This principle prescribes keeping interfaces as small as possible, creating so called role interfaces that just define one method. This avoids situations in which the change of one part of an interface forces each of its clients to change as well, even though they might not even be addressed by that change.

5.3.3 Modules

The package consists of a set of modules that together make up a simulation run. The modules are categorized by their role in the simulation run, e.g.,initializing the run, simulating model behavior, or computing/transforming the graph. Between the modules, we pass the network in which the agents live and all their feature

and attribute distance information contained in a NetworkX graph object (Hagberg, Schult, & Swart, 2008). Minimizing the information that is passed between modules makes it easier to maintain compatibility of different modules.

Here, we describe the role of the different modules, and refer to the realizations available in defSim v0.1. For a more in-depth description of the different realizations of the modules, see the software documentation.[1]

network_init The task of the *network_init* component is to create the NetworkX graph object around which the rest of the simulation revolves. It contains two main functions for producing the network: *generate_network* and *read_network*. *generate_network* can generate three commonly used graphs (ring, spatial random graph, and square lattice), but will also accept the name of any of the 130 graph generators in the NetworkX graph generator library. The function *read_network* will produce a network from a user-defined adjacency matrix or edgelist.

agents_init The *agents_init* component is responsible for initializing agent attributes. Currently, there are methods implemented that initialize F number of continuous agent features (all $o_{iF} \in [0,1]$) or F number of categorical features, all with Q number of traits. For continuous features, methods are available to generate correlations between multiple features and to create similarity to network neighbors based on one feature. The module contains the ABC *AttributesInitializer*. Implementations of the AttributesInitializer could be used to set all kinds of mutable attributes (e.g.,opinions, attitudes) and immutable attributes (e.g.,demographic information). Combinations of categorical and continuous attributes are possible, but require an appropriate dissimilarity calculator.[2]

network_evolution_sim The network evolution component contains *NetworkModifiers* which can be used to alter the structure of the physical network (e.g.,by adding or removing ties between agents). The network modifiers are applied after agent initialization and can thus use agent attributes as criteria for network changes. Currently, the *network_evolution_sim* component contains the *NetworkModifier* ABC and the *MaslovSneppenModifier* which applies Maslov-Sneppen rewiring (Maslov & Sneppen, 2002).

focal_agent_sim The *focal_agent_sim* component is the first component involved in the simulation process, and its only responsibility is to pick the focal agent. Currently, only random selection is implemented, where any one of the agents is selected with a uniform probability. The *select_agent* method from the ABC FocalAgentSelector takes the NetworkX graph and a list containing the (sub)set of agent labels from which you want to sample, and returns the index of the chosen agent. Depending on the communication regime, the selected agent can either be

[1] Available at defsim.github.io/defSim

[2] In the current version there is no DissimilarityCalculator that supports a mixture of both types.

the source (under one-to-one and one-to-many) or target (under many-to-one) of influence.

neighbor_selector_sim This component is responsible for choosing the communication partner(s) of the focal agent. Most notably, homophily—a tendency to more likely interact with similar others—has a central role in many social-influence models (Carley, 1991; Hegselmann & Krause, 2002). In the current version it is possible to either select the neighbor(s) randomly or based on their proximity in social distance based on a homophily threshold.[3] The ABC *NeighborSelector* defines the *select_neighbors* method that receives the index of the focal agent and the communication regime under which the agents operate, and returns the indices of the selected agents in a list.

influence_sim The influence component is responsible for exerting influence between the set of agents chosen for interaction by changing their opinion or belief profile according to some updating rule. It certainly has the most important task in the process and the most extensive interface. Currently, there are four different influence functions contained in the component: bounded confidence (following e.g. Deffuant et al., 2000), similarity adoption (following Axelrod, 1997), weighted linear influence (inspired by e.g. Jager & Amblard, 2005; Mäs et al., 2014), and persuasion (following the implicit argument exchange implementation by Feliciani et al., 2020). The module contains the *InfluenceOperator* as an abstract base class which defines the *spread_influence* method. This method receives the indices of the focal agent and selected neighbor(s), a list of mutable attributes, the communication regime, a dissimilarity calculator that will be used to update changed distances, and a key-worded argument (kwargs) dictionary for potential parameters of the concrete influence function.

dissimiliarity_component The *dissimilarity_component* is concerned with calculating the overlap between agents and storing their dissimilarity in the edge attribute 'dist'. defSim version 0.1 includes Hamming distance (commonly used for categorical attributes), normalized Manhattan distance, and normalized Euclidean distance (used for continuous attributes). The method *select_calculator* creates an instance of one of the calculators, that can then be used to calculate dissimilarity between agents during initialization and updating altered distances during the simulation. The ABC *DissimilarityCalculator* defines two methods: one for calculating the dissimilarity networkwide (i.e. between all pairs of agents), and one for calculating the dissimilarity between two agents. Calculating the dissimilarity is computationally expensive, so its use should be reduced wherever possible.

[3]Note that homophily can also be implemented in the influence_sim component. Where one implements homophily is not trivial, but ultimately depends on the conceptualization of homophily that the user pursues.

tools The tools component includes all modules that support the simulation in some way. The *NetworkDistanceUpdater* contains the function *update_dissimilarity* which receives a list of nodes and a dissimilarity calculator, and then uses that calculator to recalculate the dissimilarity between the agents in the list and all of their neighbors. This function is usually called by the *InfluenceOperator* after the feature vectors of one or multiple agents changed.[4] Furthermore, the module contains the *check_dissimilarity* function which can be deployed to check if the simulation reached an equilibrium. This function is called by the *Simulation* class to check for convergence. The module *CreateOutputTable* handles the generation of output for the user. In principle, it measures and returns the output at the end of the simulation run, but the user is allowed to record anything at every (or any) iteration as well. The method *create_output_table* takes a list of realizations of output measures included in defSim and/or user-written instances of the ABC. Each output is generated at the end of every simulation run. If simulations run as part of an experiment, the output from the individual simulations is combined into a single output table when the experiment has completed. The module *ClusterExecutionScript* is called by the *Experiment* class when the user wishes to run a large simulation on a high-performance computing cluster which uses the SLURM resource management system (Yoo, Jette, & Grondona, 2003). This module generates scripts which can be used to distribute simulation runs across available nodes and cores. Use of such a computing cluster can significantly reduce the time required for large-scale simulation studies. Finally, the module *ConvergenceChecks* implements several convergence checks along with the ABC from which these are derived. The convergence checks are called at an adjustable interval during the simulation run to determine whether the simulation has converged.

5.3.4 *Simulation* and *Experiment* classes

defSim is a package that consists of multiple separate modules, which makes the package highly flexible and adaptive. Most users, however, will at some point use the functions from one of two classes that handle calling all the modules for you: the Experiment and Simulation classes. With these classes, defSim users can run a complex social-influence simulation using only two lines of code:

```
1 my_simulation = defSim.Simulation()
2 my_results = my_simulation.run()
```

These lines create (line 1) and execute (line 2) a Simulation object, using all of its default values. Unaltered, they generate one run of the experiment from Axelrod (1997) with 49 agents on a lattice network who each have five beliefs with three

[4] This way, we minimize calling this computationally expensive method, by calling it only when updates are actually required instead of after each interaction.

possible traits for each feature. An experiment object—a set of simulation runs replicated a desired number of times—can be called in the same way, changing the call to the *Simulation* class to the *Experiment* class.

The *Simulation* class calls all modules at the right time, checks convergence, and returns the output once the convergence criterion is satisfied. A visual representation of what a simulation run looks like is given in Figure 5.2. All arguments that the user wishes to pass to the modules are given as keyword arguments or dictionaries. For transparency, we added the function *return_values* that returns the settings (default and user-given) of the Simulation object.

The *Experiment* functions as a wrapper around the Simulation class. It takes the same arguments, but the user can supply lists of values that will be combined to form the fully crossed set of unique conditions. Alternatively, the Experiment function will also accept a batch of simulation run objects for more complex co-variation of parameters. In addition to specifying the conditions, the user also supplies a value for the number of repetitions. Each unique condition is then run the desired number of times. Furthermore, the method *run* from this class contains the option to automatically distribute the runs over multiple processing cores, considerably lowering the execution time.

5.3.5 User-written extensions

defSim's architecture was specifically designed for the user to go beyond the package's default settings. Though the package might be interesting for hands-on teaching about social-influence models, most users will at some point like to model their own deviation from the literature. There are two ways to do this: by writing realizations of defSim's ABCs, or by using defSim's modules independently throughout your code.

Writing realizations of the abstract base classes is fairly straightforward. The ABC prescribes what it needs as input and output within that component, and by inheriting from the ABC, Python recognizes your own module as an integral part of the software. The user is then able to pass their realization to the *Simulation* or *Experiment* function directly. Furthermore, by publishing your realization, you contribute to the body of replicable, open source social-influence models, and create easy access to - and attention for your work. Simple tutorials of how to write your own realization of an ABC are available on the defSim Github wiki.[5] defSim also comes with a test suite. Any new addition should ideally also add to this test suite, to check its expected behavior and compatibility with existing modules.

The second way is by using defSim components independently throughout your code. This method is suitable for experienced Python and NetworkX users. Using

[5]The developers wiki can be found at https://github.com/defSim/defSim/wiki

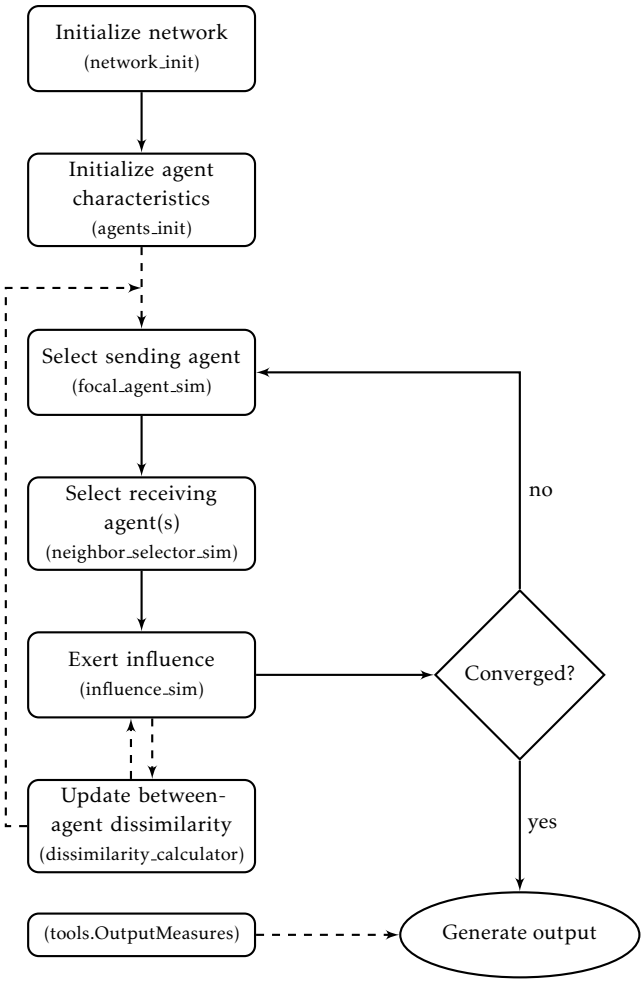


Figure 5.2 defSim modules involved in the flow of a simulation run

components of defSim this way simplifies your code and could be useful when quickly sketching a novel idea.

5.4 defSim in action: One-to-many communication in discrete event social-influence models

To illustrate the usefulness of a general simulation framework for social-influence models, we replicate the findings from Chapter 3 and test their generalizability through a comparison of two fundamentally different opinion models. The mechanism relates to an assumption that the overwhelming majority of discrete event social-influence models make: communication between agents happens on a one-to-one basis. The implications of this assumption, as argued in Chapter 3, are more impactful than the modeling tradition had (implicitly) assumed as the choice of communication regime can play a pivotal role in the emergence of fragmentation and isolation. We discuss the roots of this assumption and translate the model—originally implemented aligned with Axelrod's model for the dissemination of culture (1997) that operationalizes opinions as categorical features—to the family of models that use continuous opinions: the bounded confidence model (Deffuant et al., 2000; Hegselmann & Krause, 2002). A discussion and analysis of the similarities between the two conceptualizations of belief systems contributes to much needed knowledge about the conditions under which mechanism within the two systems generalize (Flache et al., 2017).

5.4.1 Communication regimes in social-influence models

Modelers of social-influence processes have to make important choices about when communication happens between which agents precisely. We label the different variations *communication regimes*. Traditionally, social-influence models have used simultaneous updating (e.g. French, 1956; Friedkin & Johnsen, 2011; Hegselmann & Krause, 2002). In models with simultaneous updating, the agents update their opinion in response to all available information in their network at once.

In an attempt to explicitly model the interaction events in which influence occurs, discrete event social-influence models appeared that use sequential updates between sets of agents (Axelrod, 1997; Deffuant et al., 2000). Modeling communication events that happen by chance effectively adds stochasticity to the model, making them less easily tractable, but—arguably—more realistic. Creating meaningful agent-based models is a balancing act between simplicity and accuracy. To create a more accurate representation of social-influence dynamics, models with sequential updating open the door to event-based mechanisms and explanations.

Naturally, agent-based models that exploit the simplest operationalization of the units involved in influence events focused on modeling interactions on the dyad. The logic is straightforward: at each point in time, two agents meet and exchange opinions. The significance of this largely undisputed assumption is not obvious at first. After all, it seems reasonable to reduce even the largest group interaction to a series of one-to-one communication events. Rigorous analysis of an alternative to one-to-one communication, however, showed that the interpersonal influence assumption has profound implications for model dynamics. Diversity equilibria in Axelrod's model are prone to collapse rather quickly when even the smallest amount of perturbation noise—idiosyncratic switching of beliefs—or interaction noise—incidental partner choice errors—is added. Communication regime appears to be the root to the fragility of diversity states in Axelrod's nominal model (Flache & Macy, 2011a). Flache and Macy (2011a) show that by switching from interpersonal to social influence, Axelrod's core mechanism of 'global diversity through local convergence' becomes considerably more robust to both kinds of noise, as well as to scaling effects. The model can be seen as the ABM interpretation of models with simultaneous updating, because those models already (implicitly) assumed that an individual's opinion is affected by all available information in that individual's local network (French, 1956; Harary, 1959; Abelson, 1964). Influence in Axelrod's model, according to Flache and Macy, could be modeled *socially* as a preference to adopt the modal trait on a given feature in the subset of 'influential neighbors' for any individual. At each timestep in the model, a many-to-one interaction occurs where all agents connected to the receiving agent are assigned to the set of influential neighbors with a probability equal to the share of common traits on all features between the agents. Essentially, this maps the procedure of Axelrod's homophily-based influence onto the desired *social* influence condition.

5.4.2 Interaction in online social media: the one-to-many regime

Chapter 3 highlighted the critical role of interaction regimes as a seemingly innocent micro-level assumption for macro-level phenomena in the context of online social media. The point of departure for this model is the observation that communication in online social media is naturally structured in a way much different from the interpersonal communication typical for offline settings. Online, messages are spread to many people at once, through posting on a public profile as status updates or tweets, or by talking to followers of a certain topic, for example, through hashtags or community postings. Surely, if we want to apply the models from the social-influence tradition to understand the differences between online and offline communication,

analyzing the micro-assumption of interpersonal influence will be a good place to start.

Chapter 3 investigated how *one-to-many* communication affects social-influence dynamics of cultural diffusion in Axelrod's seminal model (1997). At each point in time, an agent is chosen to send a message reflecting its opinion position or belief to all the agents in their local network. Those agents adopt the message with a probability equal to their similarity. One might think that all this effectively does is speed up the model convergence. The one-to-many interaction event, in fact, is merely a series of one-to-one interactions from the same sender. However, the circumstance that the sender of the message emits to an entire section of the network at once creates an externality for those agents who happen to *not* accept the message: they are at higher risk to become an outlier in their network than in the case of one-to-one communication. In turn, these outliers more quickly lose the similarity necessary for communication to their direct contacts and isolate more easily. Castellano et al. already observed that interaction in Axelrod's model can result in increasing dissimilarity, and that "[...] when the number of neighbors is larger than 2, each interaction can, somewhat paradoxically, result in an increase of the general level of disorder in the system." (2009, 614). One-to-many interaction speeds up this process locally, contributing to the emergence of stable diversity globally.

For small networks, Chapter 3 provided a mathematical proof of increased likelihood of cultural isolation for some agents under one-to-many communication, using Markov chain analysis. For larger networks, this method is not feasible, but agent-based simulation showed that the communication regime's tendency to produce isolation, and considerably less convergence to a single cultural profile overall, persist.

The core argument presented in Chapter 3 is more general than the model which was eventually implemented. It can be broken down into a few simple propositions:

1. Online social networks are characterized by one-to-many communication, whereas offline interaction is characterized by one-to-one communication

2. The extent to which an individual is influenced by a communicated message depends on their similarity to the sender

3. By reaching many individuals at once, one-to-many communication makes influenced individuals more similar to each other while simultaneously increasing their dissimilarity to others who are not influenced

The general argument makes no reference to specific network structures or types of attributes to which the argument applies. Chapter 3 explored several variants of the model, both to investigate substantive questions and to assess the robustness of

their results. Fundamentally, however, these extensions remain within the framework of Axelrod's model of cultural dissemination.

5.4.3 Models

Nominal model

Chapter 3 studied the emergence of consensus, clustering and isolation under one-to-many communication by building on Axelrod's seminal model for the diffusion of culture (1997). The *similarity-adoption* (SA) model with nominal opinions was precisely aligned, to ensure that its results are directly comparable to Axelrod's initial model, as well as to all later contributions that have built upon it.[6]

In the initialization stage, N agents are generated and put on an $L \times L$ grid where each agent is connected to its eight nearest neighbors (i.e. *Moore* neighborhood). Agents on the edges or corners of the grid are connected to the agents on the opposing side of the grid, creating a torus-like network structure. This final step makes sure that every agent has the same number of contacts, and, more importantly, that no agent has different structural network positions in, for example, terms of centrality or path length to other agents. Each agent is initialized with a cultural profile C, a vector of length F features with Q possible traits. These features represent topics for beliefs for which the held trait is an expression of a specific belief. In the original publication, the cultural profile C is interpreted as a set of beliefs or tastes. For example, C could consist of F topics like favorite book, music preference or fashion style, for which each agent holds one of Q tastes. Culture, according to Axelrod, has to be interpreted broadly here, as C could represent "[...] things over which people influence each other [...]" (Axelrod, 1997, 204). In the opinion dynamics literature, the vector is considered to represent a set of (political) beliefs like whether or not a social issue f demands governmental action or which q_f out of Q policies would be most effective.

In the simulation stage, a series of discrete steps is executed until the whole system is in equilibrium. At each timestep t, an agent is picked at random, to act as the agent who will try to influence its neighbor(s). Under Axelrod's rules, an interaction

[6]The model deviated in three ways, and all deviations were checked for whether they meaningfully impacted the model dynamics. First, Moore instead of Von Neumann neighborhoods were used, because Von Neumann neighborhoods are not transitive. Second, the originally used square lattice was replaced with a lattice that has no agents on the sides and corners of the lattice with fewer neighbors. All agents residing on the sides of the network are linked to the agents on the other side, creating a torus-like network. This change avoids irregularities that result from the fact that agents on the boundaries have fewer neighbors than other agents, and thus improves the parsimony of the model. Thirdly, in the original model, the agent who will be the receiving party in the interpersonal influence event is picked first, whereafter one of its neighbors is picked to exert influence. Keijzer et al. reverse this chain of events and pick the sending agent first. In networks where all agents have the same degree, both implementations are equal. In networks with uneven degree distributions, however, the procedure of picking the sending agent first ensures that every agent is picked to send a message with the same probability.

partner j is picked from the set of neighbors of i and a feature F is elected for possible influence on which i and j hold different traits. Then, with a probability equal to their proportional similarity, j adopts the trait q_{if}. This implements the *homophily* principle, the principle that more similar agents are more likely to interact (Carley, 1991; McPherson et al., 2001). One-to-many influence works in precisely the same way, except now all neighbors of the sending agent i are immediately selected for interaction and adopt the trait q_{if} with a probability equal to their proportional cultural overlap with i. The formalization of this procedure is described in Table 3.1 in Chapter 3.

From nominal to metric models

As discussed Section 5.3.1, the opinions of agents in social-influence models are typically modeled as a set of either *nominal* (i.e. categorial or qualitative) or *metric* (i.e. continuous or quantitative) features. Conceptually, there is no strong reason to favor one operationalization over the other to represent opinions. In the social-influence literature, opinions are viewed as an umbrella term that covers any kind of belief, attitude, view, conviction or judgement held by an individual (Flache et al., 2017). A nominal opinion could, for example, represent which of a series of policy alternatives an individual finds most fitting, or which of musical style they prefer to listen to. The metric opinion, on the other hand, could denote the degree to which one thinks a specific policy measure is suitable, or to what extent someone likes jazz. Some have also argued that nominal opinions are merely simplifications of the underlying continuous scale (Nowak, Szamrej, & Latané, 1990).

Direct comparisons of nominal and metric models for opinion dynamics are scarce (though some valuable contributions do exist De Sanctis & Galla, 2009; Flache, 2018b; Flache et al., 2006), and with good reason. Direct comparisons are not so straightforward as one might think. A strictly aligned and direct comparison to Axelrod's model with metric features in connected graphs knows only one equilibrium: consensus. When randomly distributing agents on some opinion dimension, it is extremely unlikely that even the most dissimilar agents will have non-zero overlap; therefore they will ultimately converge in their opinions, however long it may take. To create disorder in the system, one has to introduce (a fair amount of) noise (De Sanctis & Galla, 2009) or a similarity threshold (Flache, 2018b; Flache et al., 2006), much like the bounded confidence model for discrete event social-influence models (Deffuant et al., 2000).

Metric model

Here, we aspire to compare the effects of one-to-many interaction that were derived in the Axelrod framework to effects in an otherwise comparable model using continuous opinion spaces. We start with a seminal model for continuous opinions: the *bounded confidence* (BC) model (Deffuant et al., 2000; Hegselmann & Krause, 2002). We try to stick as closely to the initialization and simulation procedures in the SA model explained above. The implementation of BC deviates from Axelrod's model in three ways: the opinion profile possessed by the agents, the operationalization of the homophily principle, and the effect of successful interaction.

In the BC model, the cultural profile C that agents possess is replaced with an opinion vector C that contains F opinions o, each with a value $o_{if} \in [0,1]$. The starting values are drawn from a continuous uniform distribution.

The homophily principle is a core assumption in both SA and BC models, but operationalized somewhat differently. With BC, rather than letting the success of the interaction depend on the similarity between the agents, we define a threshold, a minimum degree of similarity, for interaction. By convention, the *confidence bound* ϵ is defined as the maximal allowed dissimilarity and is, thus, the inversion of the interaction probability from the categorical model $p_{ij} = 1 - \epsilon$. The dissimilarity on which the confidence bound operates is—much like in SA—a function of agent similarity on all features, such that

$$d_{ij} = \frac{1}{F} \sum_{f=1}^{F} |o_{if} - o_{jf}| \tag{5.1}$$

When agents are paired for interaction, a sending agent j influences a receiver i. With SA, successful influence leads to adoption of the given trait by i. BC adds the option of smoothing the adoption with an influence weight. The convergence rate $\mu \in [0,1]$ sets the proportion of the distance that the receiving agent will move towards the opinion of sending agent i, such that

$$o'_{if} = \begin{cases} o_{if} + \mu(o_{if} - o_{jf}), & \text{if } d_{ij} \leq \epsilon \\ o_{if}, & \text{if } d_{ij} > \epsilon \end{cases} \tag{5.2}$$

Conceptually, there are two obvious ways in which SA with $F = 3$ and $Q = 2$ can be operationalized in the metric model. One could argue that the number of dimensions should stay the same, and the convergence rate should be set to 1, matching the complete adoption like in SA. This operationalization would match the categorical model closely, but at the expense of granularity in the starting values. The continuity of the opinion scale is more or less ignored by the influence function, because any successful interaction results in one of the two agents abandoning their position.

With pure assimilation, the number of unique opinion positions on the continuum will strictly decrease over time.

A second scenario worth investigating is the unidimensional opinion case, well understood in the BC tradition. Considering the impact of successful communication on the agent-similarity matrix, we can observe that in the nominal model with $F = 3$, every successful interaction increases the similarity with $1/3$ of the total potential distance. The equivalent in the BC world with $F = 1$ would be to set the convergence rate parameter μ to $1/3$. The downside to this operationalization is that the convergence rate in the BC model is not proportional to the total possible distance between agents, but, rather, proportional to the existing dissimilarity between two given agents. Where successful interaction between two agents with a dissimilarity score of $1/3$ in SA would result in completely similar agents, the BC model results in a dissimilarity between the agents of $2/9$.

Finally, we need to define a confidence bound for both scenarios. In the SA model, there is no explicit confidence bound. However, implicitly, this confidence bound exists at $(F - 1)/F$. Beyond this point, agents share less than one trait on a feature, so p_{ij} drops to zero. A confidence bound of $2/3$ is, however, not a viable option as the tendency towards monoculture overpowers any other dynamic, and differences that may exist between conditions do not manifest in equilibrium. We define the confidence bound in both BC models to be half the value of the implicit confidence bound in SA, setting it to $1/3$.

Software implementation

Replicating the experiment from Chapter 3, and translating its main findings to categorical opinion spaces, can be done using the default functionalities of defSim. The code to create and run the bounded confidence experiment with $F = 3$ is presented in Algorithm 5.1. The code to replicate the analysis from Chapter 3, and the replication in the metric opinion space with $F = 1$ are available in a dedicated repository.[7]

5.4.4 Analysis

For each condition we ran 1,000 independent realizations of the simulation model and stopped when they reached an equilibrium. The SA model was considered to be in equilibrium when all agents that share a connection have either completely identical or completely dissimilar cultural profiles. When the entire model is in that state, no further interactions are possible. The BC model is in equilibrium when all connections link agents with approximately equal opinions (a threshold

[7] The repository can be found at github.com/marijnkeijzer/o2mNominalMetric

Algorithm 5.1 Code for the bounded confidence model with $F = 3$. All parameters that are called in the execution of the experiment are specified, even when they are not required by the Experiment function as they have functional default values.

```
experiment_bc_3f = ds.Experiment(
    seed = 314,
    influence_function = "bounded_confidence",
    attributes_initializer = "random_continuous",
    attribute_parameters = {"num_features": 3},
    dissimilarity_measure = "manhattan",
    topology = "grid",
    network_parameters={
        "num_agents": [4, 9, 16, 25, 36, 49, 64, 81, 100],
        "neighborhood": "moore"
    },
    influence_parameters={
        "confidence_level": 1/3,
        "convergence_rate": 1
    },
    communication_regime = ["one-to-one", "one-to-many"],
    max_iterations = 100000,
    stop_condition = "strict_convergence",
    stop_condition_parameters = {
        "convergence_dissimilarity_maximum": 1/3,
        "convergence_dissimilarity_minimum": 0.01,
        "cluster_dissimilarity_threshold": 1/3
    },
    output_realizations = ["Basic", "Isolates"],
    repetitions = 1000
)
results = experiment_bc_3f.run()
```

for maximally allowed dissimilarity of 0.01 was chosen), or with opinions that fall outside of the confidence bound, in which case there is no further influence possible either.

Following the approach from Chapter 3, the outcomes of the runs are described by whether they ended in monoculture and whether a run produced at least one isolate. SA of opinion dynamics tend towards monoculture when the size of the population is increased. However, one-to-many interaction has proven to be robust to this general tendency, primarily by creating isolates—agents that do not possess any overlap with their connections that would allow them to successfully interact. These two descriptive statistics therefore indicate whether this tendency persists in models with continuous opinions. What is more, their difference can be interpreted as the proportion of runs that created one or more clusters of any size between 1 and N.[8]

[8]This is not to say that such clusters cannot exist in runs with at least one isolate as well. A thorough analysis of cluster sizes in Chapter 3, however, showed that clusters of size 1 (i.e. isolates) are between ten to a hundred times more likely to occur in such models than any other particular size.

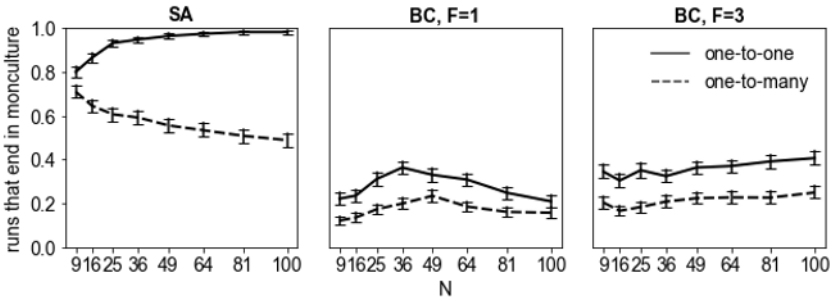


Figure 5.3 Share of runs that end in monoculture by communication regime and size of the network for each of the three models. Each datapoint represents 1,000 independent realizations of the simulation model, error bars indicate 95% confidence interval

Figure 5.3 describes the equilibria in terms of proportion of monoculture in the three models of interest. The first panel clearly replicates the results from Chapter 3. In small networks, communicating to all neighbors at once decreases the tendency to converge to a single, shared opinion profile. As the size of the network is increased, convergence becomes increasingly likely with one-to-one interaction—an observation that dates back to the original rendition of the model by Axelrod (1997)—but *less* likely with one-to-many communication. This resonates, too, with the findings of Flache and Macy (2011a), who find that many-to-one influence produces more diversity in larger populations. In one-to-many influence, most of the runs without monoculture will produce resilient clusters of sizes between one and N, with no noticeable probability peak at any point, but a relatively large number of runs will end with at least one isolate. The first panel of Figure 5.4 shows that between 10 and 15 percent of the runs with one-to-many interaction produce one or more isolates, where the share of the runs with an isolate in the SA with one-to-one influence is close to zero as the size of the grid is increased.

Looking at the two BC models, there are a few noteworthy differences and similarities. To start with the latter, the tendency of one-to-many communication to produce lower numbers of runs that end in monoculture appears to replicate. The likely explanation for this is that those agents who are not influenced by a sending agent will be quicker to lose active links—links that connect agents with a similarity score within the confidence bound—to mutual contacts.

It is also apparent that the differences between the communication regimes manifest in different ways. First of all, the proportion of monoculture in metric models is considerably lower than in nominal models. To understand why that is the case, consider two agents that are connected through an inactive link. In the nominal model, any change in their opinion profile will reactivate this link, without exceptions.

After all, complete dissimilarity is a unique configuration when $Q = 2$. In the BC model, reactivation of the link is rarer because it is possible that opinion changes are not large enough to move the agents into each other's confidence bound. This only happens when one of the two agents is influenced by an agent whose opinion is, of course, within the confidence bound of the other, and who incurs an opinion shift large enough to pull the opinion of the receiving agent into neighboring agents' confidence bounds.

Secondly, the size of the network has a less clear effect on the probability of ending up in monoculture. In the nominal model with one-to-one communication, it is known that the size of the grid generally increases the amount of homogeneity. Since any successful interaction in the nominal model leads to distance minimization between the agents, anything that increases the amount of interaction will lead to homogeneity overall (Castellano et al., 2009). As the size of the network increases, clusters that are in the process of creating stable boundaries locally, run the risk of encountering a different trait propagating as a wave through the network, loosening the boundaries and creating the possibility of convergence between previously stable dissimilar clusters. With one-to-one interaction this risk is high and increases with network size, because the time period is much longer in which clusters are not fully crystallized out yet and are thus susceptible for adopting "foreign" traits, compared to one-to-many interaction. With one-to-many interaction, the quick stabilization of clusters instead limits their size so that larger networks can contain more clusters and are less prone to develop monoculture.

The metric model exhibits a more complex relation between the communication regime and the effect of network size. Panels b and c of Figure 5.3 show that the share of runs ending in monoculture increases with network size in at least a part of the parameter space also under many-to-one communication, and it decreases in another part of the parameter space under both communication regimes when $F = 1$. Explicating the reasons for this more complex link in detail goes beyond the scope of this chapter. We suspect that there are two key differences between the BC and the SA model when it comes to the robustness of emergent clusters, which jointly generate the complex pattern of effects in the BC models. First, as discussed before, the reactivation of links through encounter with foreign traits is a rarer circumstance in the metric opinion space of the BC model. Second, once a link is reactivated in the BC model, the linked agents influence each other with certainty regardless of their exact similarity, while in the SA model this influence remains a probabilistic event. While the first difference tends to make clusters more resistant against encountering more outside influences in larger networks, the second difference works towards making clusters less resistant under this condition. In the conditions we inspected, we found that the interplay of these processes adds up to the effects of increasing the

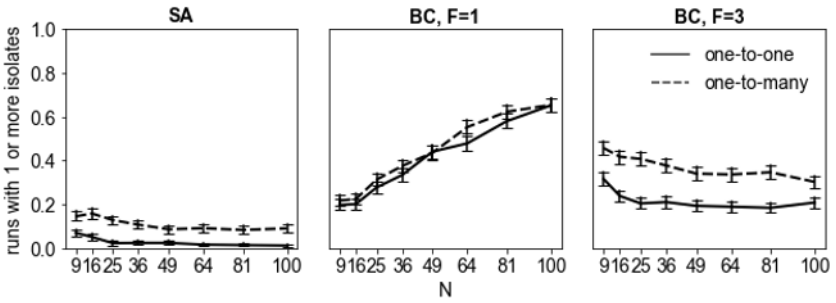


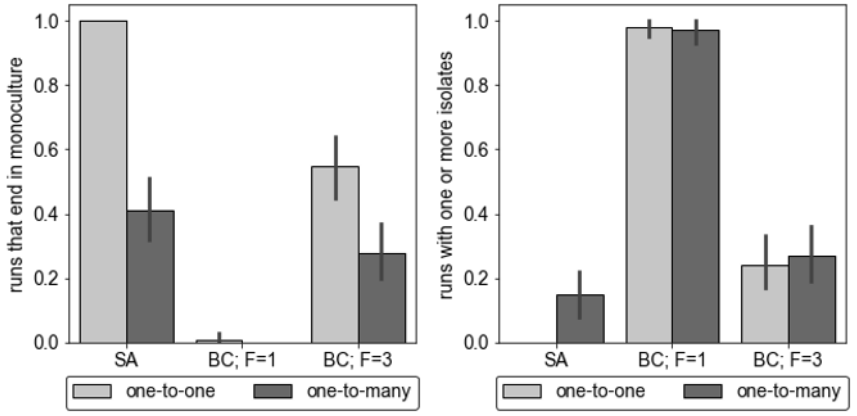
Figure 5.4 Share of runs that end with one or more isolates by communication regime and size of the network for each of the three models. Each datapoint represents 1,000 independent realizations of the simulation model, error bars indicate 95% confidence interval

size of the network being less pronounced in both metric models compared to SA. Most importantly for our question, the differences between the two communication regimes do appear consistent and stable across different network sizes, albeit much smaller than in the nominal world.

When we zoom into the number of runs with an isolate shown in Figure 5.4—a special case of the non-monoculture equilibrium—a couple of interesting differences appear between the nominal model and the metric models in general, and between the metric models themselves. In models with continuous opinion dimensions, the number of isolates is higher in all conditions. Again, this is likely the result of the fact that reactivation of inactive bonds is less prevalent in metric models. Once local clusters have converged to a consensus, any isolated agent will remain detached.

The pattern of between-communication-regime differences and size of the grid appears similar in the categorical and the continuous world with $F = 3$. This is, however, not at all the case for the unidimensional metric opinion model where no between-regime difference is found. Moreover, the number of isolates increases linearly with the number of agents. The decreasing likelihood of monoculture and an increasing probability of finding any isolates in equilibrium underlines the idea that the unidimensional model is much less affected by the size of the grid. The dynamics of opinion formation will unfold at a local scale, and the regions affect each other less.

The apparent decreasing tendency towards monoculture in the metric model with $F = 1$ and the convergence of the number of isolates in the metric model with $F = 3$ suggest that differences may not persist in larger populations. To that end, we quadrupled the size of the network and simulated until convergence. For each condition, one hundred independent realizations were run with networks of size $N = 20^2$ and their equilibria are described in Figure 5.5.



(a) monoculture **(b)** isolation

Figure 5.5 Share of runs that end in monoculture (panel a) and share of runs that end with one or more isolates (panel b) by communication regime for each of the three models. Each datapoint represents 50 independent realizations of the simulation model, error bars indicate 95% confidence interval

In larger networks, the differences with regards to the likelihood of monoculture for the two communication regimes persist. In the metric model with $F = 1$, monoculture becomes so unlikely, that the agents in all but one run in the one-to-one condition did not converge to a single opinion. As a consequence, the number of runs that produce at least one isolate is close to one in either condition. The metric model with $F = 3$ does exhibit between-communication regime differences that are closer to the nominal model, though the differences in proportion of runs that produce isolates is marginal.

We have discussed possible explanations for the differences found between the three models of interest that rest on local level processes. However, there is a noteworthy difference on the global level that could affect the trajectories of convergence: a similarity bias on the outset in metric opinion models. All three models were initialized with random values on the one or three features held by the agents. However, the similarity distributions created by those random initializations for metric and nominal models is quite different. In the nominal model with $F = 3$ and $Q = 2$, the opinion profile of the agents can be in one of eight states. The between-agent similarity, as a consequence, will always be 0, 1/3, 2/3 or 1. Initialized randomly, the similarity distribution will have an average of 1/2, where the similarity states of 1/3 and 2/3 are both three times more likely than the extremes 0 and 1. In the metric world, however, random initialization will provide agents with an opinion distance that is 1/3 on average, and, thus, biased towards smaller similarity distances.

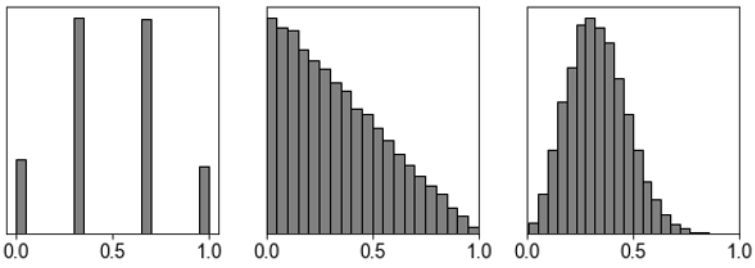


Figure 5.6 The distribution of similarity scores in the three scenarios of interest

The similarity distribution generated in fact, is one where similarity at the outset is linearly and monotonically decreasing with a median of approximately 1/3. The similarity distribution of the metric model in higher dimensions is the product of this distribution and, as F increases, will look more and more like a truncated normal distribution with a median of around 1/3. The distributions generated by the three models of interest are shown in Figure 5.6.

These differences in the similarity distributions might explain why the proportions of runs that end in monoculture are considerably lower, and isolation higher in metric models of opinion dynamics studied here. While one might expect that smaller opinion distances on the outset equates to more convergence overall, we have to consider the outliers too. As the size of the network increases, the likelihood that there are agents who do not share sufficient overlap with their neighbors will increase. However, because the average similarity is lower in metric models, the neighbors of the initially isolated agents are more likely to overlap with their neighbors, as compared to the nominal model. The probability that they integrate into the opinion cluster(s) that surround them is, thus, lower.

5.5 Discussion

This chapter introduced defSim—the discrete event framework for social-influence models—answering the literature's call to build common ground through alignment of models, and open sharing of model implementations. The past decades have produced a bulk of theory and models for social-influence models of opinion dynamics, but "[i]ronically, however, there seems to be too little influence between social-influence modelers to create the dynamic necessary to develop scholarly consensus on what are core model ingredients, in which implications models really differ and where they don't, and how models can be compared." (Flache et al., 2017, § 3.3).

The joint simulation framework we presented here is built on a set of seven principles. That is, the models in defSim (1) are agent-based, contain agents that (2) posses features and live on a (3) physical and a (4) attribute distance network and (5) exert influence on each other, (6) divide time into a series of discrete events, and (7) structure communication by a regime. It is designed using a factory method pattern, and implements adaptable abstract base classes for all the included modules. Here, we have mentioned and briefly explained all the elements included in defSim v0.1.

Version 0.1 as presented today is not complete. As others have done in this literature, we too dream of a rich model that is able to accurately identify important characteristics of complex systems, and produce hypotheses to enhance our understanding of opinion dynamics. To achieve that goal, building upon each other's implementations of assumptions is critical. If not within defSim, then in another joint simulation framework. We welcome all comments, criticism, and contributions to defSim on GitHub. GitHub offers a structure for open collaborations in software development. Current implementation of various assumptions can be discussed in openly in *issue* threads, users can contribute their own model extensions as *forks* of the latest version of defSim, and directions in which to develop the framework can be suggested as *projects*.

Using a joint simulation framework has some obvious advantages when it comes to reproducibility and extendability of a common model, but there are downsides to such an approach too. First, concentrating efforts by 'putting our eggs in one basket' may make the risk of error smaller, but their consequences much larger. We advise anyone who is interested in seriously modeling a phenomenon to replicate their findings themselves using a different programming language. Whether you use defSim or not, it is always good practice to reprogram your model as a means of checking your code. Second, alignment and adoption of a shared set of model principles may make it easy to overlook implicitly included or excluded assumptions. To that end, a thorough theoretical and/or empirical motivation, and a clear description of the conceptual model are crucial to any social-influence model extension.

While creating defSim, we noticed that some seemingly incompatible modeling paradigms are actually implicitly compatible (they are unintentionally *docked*). Agent-based models of social influence are grounded in different scientific traditions and have therefore adopted a variety of names and labels for qualitatively similar assumptions. The principles on which defSim is built (listed in Section 5.3.1) are general enough to include many other variations of social-influence models than the ones we explicitly refer to now. Docking models that are traditionally unassociated and collecting those associations in the software's documentation will create a rich repository for those who wish to learn about modeling opinion dynamics.

Where some models may prove implicitly compatible through docking, other seemingly similar models can prove difficult to compare. Direct comparison may, though, provide important insights into the generalizibility of known mechanisms at play in one of them. By exposing whether these mechanisms replicate, we gain a deeper understanding of social-influence models in the broader sense.

An interesting example of a model that is implicitly compatible is the *construct* model (Dipple, Kowalchuck, Altman, & Carley, 2021). This model was designed based on the idea that perceptions of social reality are obtained through cycles of frame-development and information-gathering (Carley, 1986). Personal views are based on facts that are either known or unknown to individuals who interact and exchange facts with the probability proportional to their relative shared basis of knowledge (Carley, 1991). Though the purest form of the construct model is technically a dissemination model, the model has a high degree of implicit compatibility with discrete-event social-influence models for it is based on principles of influence as a function of social distance.[9]

Section 5.4 showed model comparison in action using defSim. We aimed to replicate the findings of Chapter 3 and investigate their generalizability in the framework of the prominent bounded confidence model for metric opinions. The communication structure in agent-based models of opinion dynamics is a hitherto understudied topic that can have decisive impact on the equilibria that come about. Moreover, one-to-many communication, we have argued, is a more accurate representation of communication in online social media, a topic that attracted attention in the opinion dynamics literature in recent years. Here, we have shown that, under certain circumstances, the mechanism of isolation and clustering through rapid convergence under one-to-many influence is a general mechanism that affects the dynamics regardless of the measurement scale of opinions. We have seen that the emission of messages to all network neighbors at once in the bounded confidence model, leads to less consensus, and—in small and medium sized networks—more opinion isolation.

The concurrent aim of this analysis was to provide an example of what the simulation software defSim can do, and how to implement a rather rich comparison with a concise piece of code. This comparison only scratched the surface of software capabilities, and what interesting comparisons can be made between the traditions

[9]Relating the construct model to the models used in this chapter, it is perhaps most similar to the bounded confidence model from Section 5.4 that used three features and a convergence rate of 1. This transformation of the BC model creates a model where only two values for each feature remain, in equilibrium. The two major differences between the construct model and that peculiar version of the BC model is that in construct (1) social distance is hamming and (2) social distance is calculated only based on the amount of shared facts about the world and not based on the amount of unknown facts. In other words, only one of two possible values of any given belief feature is used to calculate dissimilarity to interaction partners.

of nominal and metric models of opinion dynamics. Future work on comparison of SA and BC models could yield a more thorough understanding regarding the generalizability of the rich body of knowledge in both modeling traditions.

In the realm of social-influence models, there are many viable candidates for inclusion into the defSim framework. Any visible and interesting addition to the literature will benefit from increased validity through replication or from accessibility through distribution in the package. Some fruitful next steps here could be to integrate recent ideas like the addition of gossip (Deffuant, Bertazzi, & Huet, 2018), emergent coherent belief-structures (Goldberg & Stein, 2018; Banisch & Olbrich, 2019; Van der Maas, Dalege, & Waldorp, 2020), co-development of belief and physical networks (Dykstra, Elsenbroich, Jager, Renardel de Lavalette, & Verbrugge, 2013; Dykstra, Jager, Elsenbroich, Verbrugge, & Renardel de Lavalette, 2015; Edmonds, 2020), action-opinion inference (Tang & Chorus, 2019), weighted median influence (Mei, Bullo, Chen, & Dörfler, 2019), the triple filter bubble (Geschke et al., 2019), or the co-evolution of intergroup attitudes and contact (Flache, 2018a).

The growing repository of influence functions, communication regimes and agent initializers presents some future directions for research that could be considered low-hanging fruit. Many unique and interesting combinations of assumptions can be explored using defSim 'out of the box' (i.e. with a low degree of modularity). For example, two early implementations of models in defSim explored the effects of increased personalization in online social networks (Keijzer & Mäs, in press) and the effectiveness of social bots in the spreading of misinformation (Chapter 4). Furthermore, researchers could use low modularity defSim to explore well-understood, one-to-one-based models in their compatibility with many-to-one (Flache & Macy, 2011a) or one-to-many (Keijzer et al., 2018) influence regimes.

Technological advancement has put social simulation researchers' ambition of building complex and rich models within reach. The combination of growing computational power, expanding knowledge of well-grounded models, and increased attention to simulation-based research promises a bright future for the discipline. Now more than ever before, we need to work transparently, fast, and rigorously, to create a firm base of knowledge and worthy policy insights. Joint simulation frameworks that are openly shared and maintained by the community are key in achieving those goals. Let's hope that gradually our field will adapt to its time and indeed will prove to be buoyant in the age of open science.