Agent-based modeling for sustainability

Matt Turner

2025-04-13

Table of contents

Welcome!

These are the course notes and manual for the Agent-based Modeling for Sustainability, the second course in the Computational Social Science for Sustainability (CSS4S) series at the Stanford Doerr School of Sustainability.

The course and the `socmod` library are under active development. If you have suggestions, questions, want to report typos, etc., please help me out and open an issue on the repo's GitHub page.

This online book is divided in two parts:

1. Notes on the theory and application of agent-based modeling to promoting the social diffusion of sustainable practices through social learning and social influence. Continue on to the Introduction to read the Notes.
2. Technical manual that reviews the skills and techniques needed to develop agent-based models for sustainability and present the results in a scientific article.

Please select a chapter from the list to the left or continue to the Introduction.

1 Introduction

Indigenous, local peoples of the South Pacific Islands and other coastal habitats have sustainably managed mangrove forests to dissipate storm surges and prevent erosion, mitigating potential costs of climate change since long before the anthropocene (Alongi 2002; Nalau et al. 2018; Pearson, McNamara, and Nunn 2020; McNamara et al. 2020). First peoples of western North America have similarly practiced prescribed burns to prevent destructive wildfires during times of drought for millennia (Eisenberg et al. 2019; Kolden 2019). Despite their long-known effectiveness, adaptive practices like these often fail to spread widely. Instead, international development agencies frequently advocate for the construction of seawalls, for example, even though seawalls can exacerbate flooding once breached and incur high maintenance costs (Piggott-McKellar et al. 2020). Inland forest management is beset by polarization among stakeholders (Swette, Huntsinger, and Lambin 2023), resulting in devastating wildfires burning a buildup of fuels or greenhouse gas-intensive clearcuts.

Here we introduce what we dub *the puzzle of diffusion*: why do some effective, sustainable practices like mangrove forest management fail to diffuse broadly, while other less effective or even maladaptive practices become widespread (Figure 1.1)? To answer this, we answer some more basic questions along the way. First, how do people decide what to do, i.e., how does learning work? Second, what is the effect of identity on how well or how likely we are to learn from people of the same or different identity? Third, what is the effect of social structure, i.e., who knows whom, represented as a social network? Agent-based models are useful because they provide a framework for creating rigorous, mechanistic, concrete models of social diffusion of adaptations. They help us deal with the complexity of causation in the real world through strategic selection of causal input variables and model details, which, over several model time steps, lead to social emergent phenomena like the diffusion of climate change adaptations.

The focus of this course is developing agent-based models that can help us simplify complex combinations of cognitive and social factors to represent only the most relevant ones, and observe the effect of these on simulated outcomes such as the proportion of people adopting adaptive behaviors, or opinion extremism and polarization.

1.1 Why agent-based models are useful

Agent-based models (ABMs) provide a structured way to explore complex systems by simulating interactions between autonomous *agents*, i.e., simulated people. In sustainability contexts,

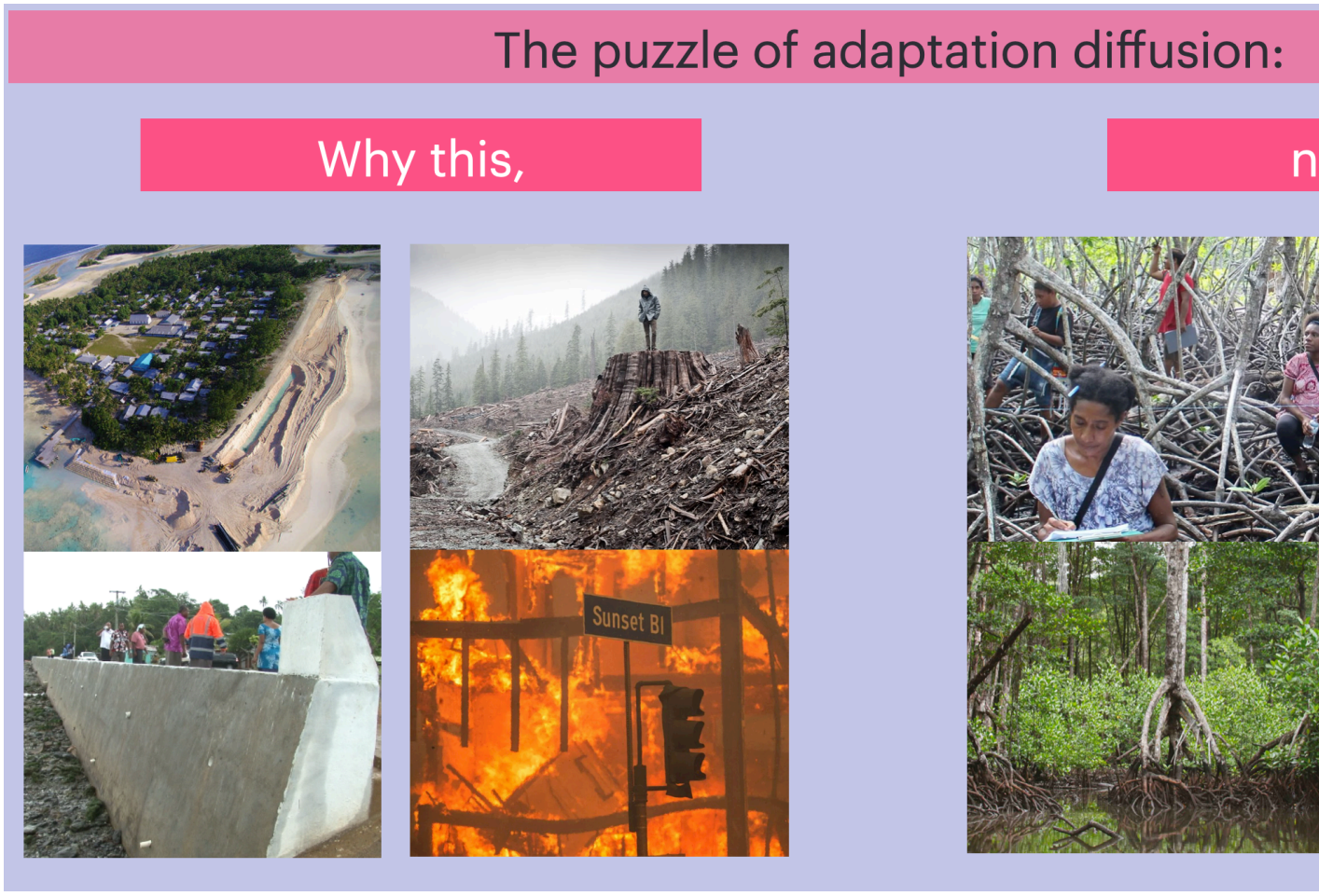


Figure 1.1: Agent-based models can help us answer the puzzle of diffusion, i.e., why do certain adaptations widely diffuse socially and some do not, with maladaptations often taking their place?

ABMs offer a low-cost testbed for understanding how interventions might impact social dynamics and environmental outcomes. For example, Airoldi and Christakis (2024) demonstrated through regression analysis across that one method for selecting individuals targeted in a public health education campaign worked better than another. They studied over 20,000 individuals across Honduran villages of about 100 people each to reach their findings. Real-world verification of the efficacy of different intervention strategies is important. However, we can also use agent-based models to represent the diffusion of information in simulated populations where interactions are structured by model social networks. We can initialize thousands or millions of simulated villages in which this information could diffuse with different intervention strategies, and observe the distribution of the adoption of sustainable behaviors for each potential intervention strategy. We can then analyze which performed best *in silico*, which can be helpful if interventions will be taken to different contexts. In other worlds, we can use ABMs to deduce how different learning rules, group identities, and social structures shape sustainability outcomes generally, which can guide our selection of real-world intervention strategies.

A typical ABM simulation cycle includes (Figure 1.2):

- Initializing agent populations and environmental conditions
- Iterative steps where agents select partners, interact, and update behaviors based on outcomes and learning rules
- Repeating these steps until specific conditions or thresholds are met
- Generating output data for analysis

We will analyze and draw on several real-world empirical studies of interventions to develop our agent-based models that we in turn will use to simulate interventions in order to *deduce* which strategies are most effective for *social interventions* to promote sustainability, and why. A *social intervention* (or just *intervention*) for promoting sustainability is any concerted effort where those promoting a sustainable practice introduces information about how to perform that practice to a population. *Deductive* methods complement regression-based inferential or *inductive* strategies. Deductive strategies can explain which strategies are most effective and why in idealized, cost-free settings (cost-free at least compared to the cost of real-world social interventions at scale).

Low-cost experimentation with simulated social interventions to promote sustainability are critical. Unless progress is accelerated towards we can expect to "have 575 million people living in extreme poverty, 600 million people facing hunger, and 84 million children and young people out of school. Humanity will overshoot the Paris climate agreement's 1.5°C 'safe' guardrail on…temperature rise. And, at the current rate, it will take 300 years to attain gender equality" (Malekpour et al. 2023, 250). Accelerated transformations are required to reach goals necessary to avoid increasingly frequent and costly climate change disasters (United Nations 2023). It is not plausible to do real-world experiments at the global scale required to infer which strategies work best in which situations.

In this course we will focus on deducing how between different cognitive and social factors, or other initial conditions, affect simulated sustainability intervention outcomes. We will frame

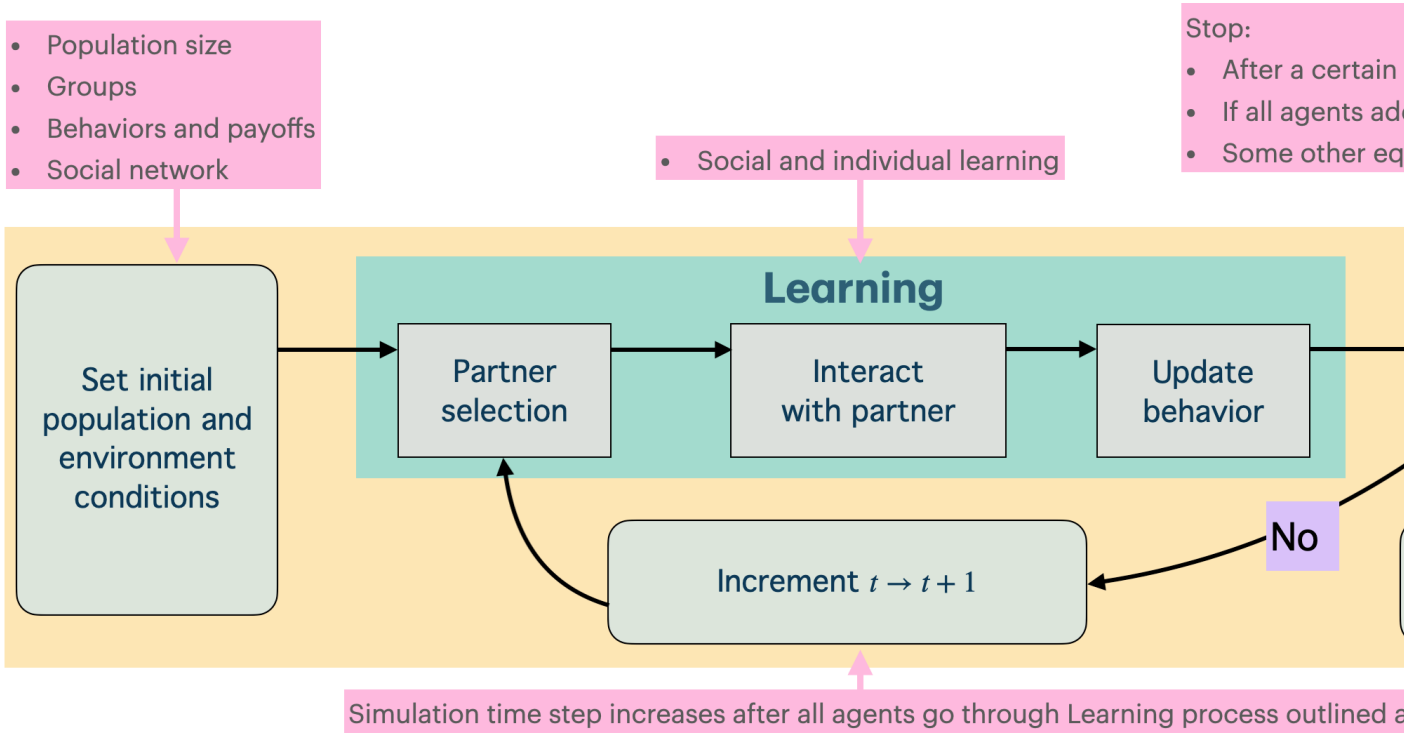


Figure 1.2: Agent-based modeling simulation cycle.

our studies in terms of sustainable development goals, but we will never fit our models to observations. Nonetheless, we will strive to develop models that are amenable to real-world interventions against which the models could be fit and predictions could be compared. It seems this is not done too much in practice yet in sustainability. However, some of our colleagues focused on studying basic processes that underlie cultural transmission do exactly this to explain experimental data and archaeological observations (Deffner et al. 2024), which thereby improves their theory, models, and understanding of cultural evolution in a theory-model-observation cycle. With more time and research effort, this cycle may become commonplace in sustainability.

The urgent need to understand how sustainable behaviors spread in order to develop effective interventions pressures social scientists to make social science more rigorous, reliable, and digestible by non-social scientists. In the rest of this Introduction we review cognitive and social theories of social learning, identity and influence, homophily and core-periphery network structures, and preview the remainder of the course material. For an overview of the course feel free to skip ahead to the Plan in table format.

1.2 Social learning strategies

Human kind is set apart by powerful learning and reasoning capabilities (Witt et al. 2024) that enable cultural transmission and accumulation of technologies and practices no other species matches (Henrich 2015). For our sustainability models, we only need simple models of cognition and learning. It would never be practical to do psychological or cognitive tests in the context of sustainability interventions that targets large populations, for one thing, so we could never compare detailed cognitive assumptions or predictions with reality. For our purposes we will consider three general classes of learning strategy:

1. **Success-biased learning**: Individuals are more likely to adopt behaviors perceived to be successful or beneficial.
2. **Frequency-biased learning**: Individuals adopt behaviors because others are already doing them, creating a conformity effect.
3. **Contagion**: Individuals copy behaviors simply by observing others performing them.

In the mangrove versus seawall example, success-biased learning might favor seawalls if influential external actors, who seem successful or wealthy, advocate for them, even if seawalls are ultimately less effective. Mangrove forests might become widely adopted if, on the other hand, frequency-biased learning predominates and many communities have adopted that method.

1.2.1 Formal social learning models with example

To make this more concrete, we *formalize* (i.e., *give formulas for*) these three learning strategies as follows using the example in Figure 1.3. In the example, there is one *focal agent* who is the

one doing the observing/learning, labelled with the ID *1*, and three social network neighbors with IDs *2, 3,* and *4*. The focal agent is deciding whether or not to install residential solar. One of his neighbors has installed it, 2. Based on 1's perception, 2 is the wealthiest, represented by four dollar symbols. 3 is perceived to have one dollar and 4 has two dollars.

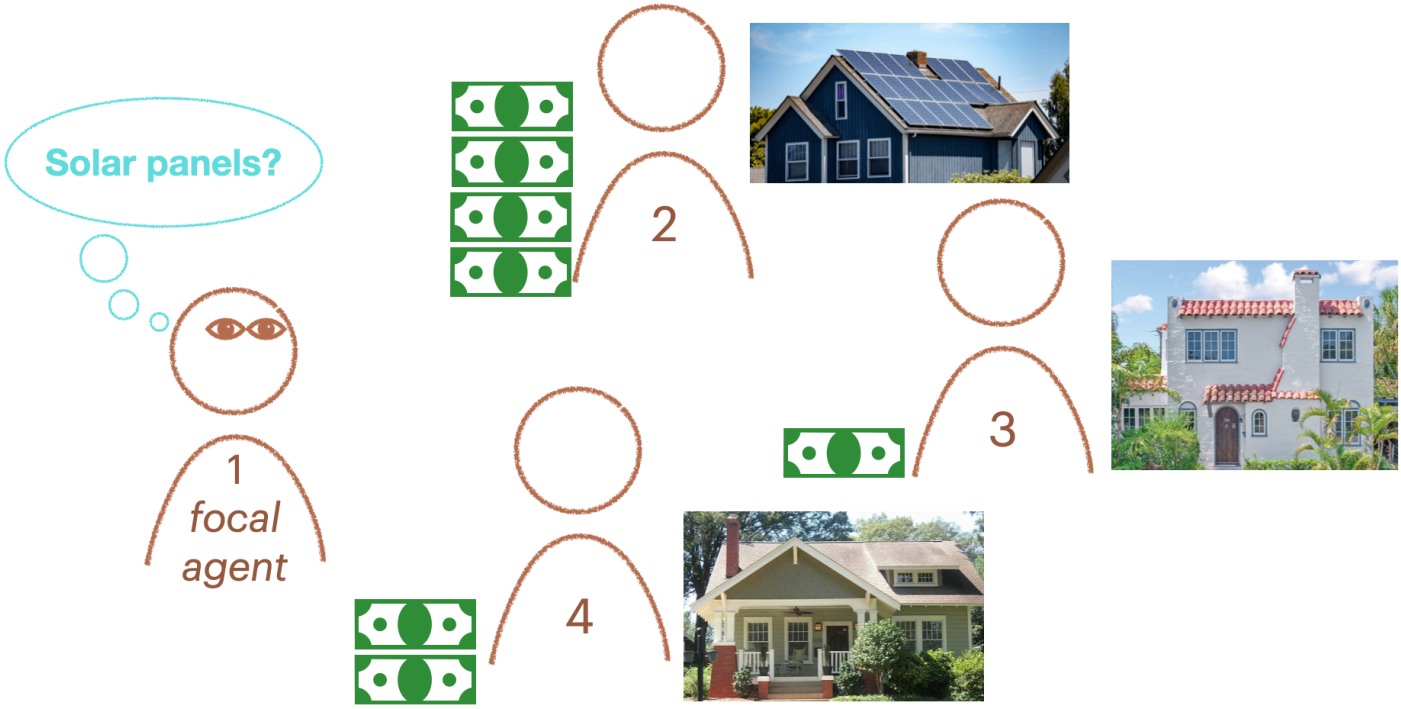


Figure 1.3: Agent 1, the focal agent, is considering whether to install residential solar. The probability agent 1 installs residential solar depends on its social learning strategy.

1.2.1.1 Success-biased learning

In success-biased learning, learners first choose an interaction partner/teacher randomly weighted by observed fitness of their neighbors, which is a generic term for wealth, power, status, etc. In general for this case, the probability that learner i chooses teacher j is

$$\Pr(i \text{ chooses } j) = \frac{f_j}{\sum_{k \in n_i} f_k}.$$

The probability that i adopts A is then the sum of the probabilities of choosing each neighbor performing A (contained in the set m_i),

$$\Pr(i \text{ adopts } A) = \frac{\sum_{j \in m_i} f_j}{\sum_{k \in n_i} f_k}.$$

In our example, only one neighbor installed residential solar with a fitness (i.e. *wealth* in this example) of 4, while the others have finess 1 and 2. Therefore, $\Pr(1 \text{ adopts } A) = \frac{4}{7}$.

1.2.1.2 Frequency-biased learning

In frequency-biased learning there is no interaction partner or teacher chosen. The probability of adoption is only given by the relative frequency of each behavior. The general expression is

$$\Pr(i \text{ adopts } A) = \frac{|m_i|}{|n_i|}.$$

In our example, then, the probability of installing residential solar under frequency-biased learning is $\frac{1}{3}$.

1.2.1.3 Contagion learning

In the contagion learning model, the focal agent's (i.e., the *learner*'s) *teacher* (i.e., interaction partner) is chosen at random. Then, the learner adopts the teacher's behavior with probability α, the adoption rate. In this case, then, the probability that 1 adopts residential solar is

$$\Pr(1 \text{ adopts solar}) = \alpha \Pr(1 \text{ selects } 2 \text{ as teacher}) = \alpha \frac{1}{3}.$$

More generally, for focal agent i with the set of neighbors n_i ($n_1 = \{2, 3, 4\}$ in the example), where m_i is the set of neighbors who have adopted the adaptive behavior ($m_i = \{2\}$ in the example). We call the adaptive behavior A. The general probability of adoption in contagion learning is therefore

$$\Pr(i \text{ adopts } A) = \frac{\alpha |m_i|}{|n_i|},$$

where the $|\cdot|$ operator counts the number of elements in a set.

1.2.1.4 Combinations of learning models

There is no reason different learning models cannot be combined. The software we will use in this class, `socmod`, provides flexibility to the user to define their own learning models. The simplest combination of the three learning models above is to add an adoption rate to either frequency- or success-biased learning. In this approach, one could call the behavior selected by the models above could be considered *prospective* or *candidate* behavior to learn, then is actually learned with probability α.

For another potentially useful modification, α could be defined at the individual level, say α_i, or for *dyads* (i.e., a pair of interacting individuals) , α_{ij}, where i is still the focal agent learner, but we have added j, representing the selected teacher.

1.3 Identity and Influence

Group identity critically influences social learning. Neuroscience research demonstrates that our brains distinctly respond to individuals identified as part of our group (Cikara and Van Bavel 2014), as revealed through fMRI neural imaging (Figure 1.4). This ability likely evolved because when humans first emerged about two million years ago, it was much more important for survival to be able to rapidly identify whether someone was a friend or foe based on group markers. Although group membership can affect how we respond to information learned from others, group membership itself is quite plastic, meaning who belongs to which group can be rapidly reconfigured. For example, neural signals of race-based group perception was observed to be suppressed and overridden when individuals were in mixed-race groups created by experimenters that competed against other mixed-race groups in an psychological experimental task (Van Bavel, Packer, and Cunningham 2008).

Studies further show that group identity can strongly influence behavioral choices. For instance, experimental evidence reveals people resist adopting beneficial behaviors if associated with opposing political identities (Ehret et al. 2022), emphasizing how identity can create substantial barriers to sustainability. This general effect of group membership interfering with learning is called *outgroup aversion* (Smaldino et al. 2017).

1.4 Social networks, homophily, and core-periphery structure

Social structure can significantly impact behavioral diffusion, especially in core-periphery configurations. Core-periphery networks emerge as a response to risk and uncertainty, e.g., in food sharing networks (Ready and Power 2018), so they are hypothesized to also be important in climate change adaptation transmission networks (Jones, Ready, and Pisor 2021). Core-periphery networks can be created by setting appropriate group sizes and *homophily* levels in homophily network models (Turner et al. 2023) or specifying certain connectivity probabilities

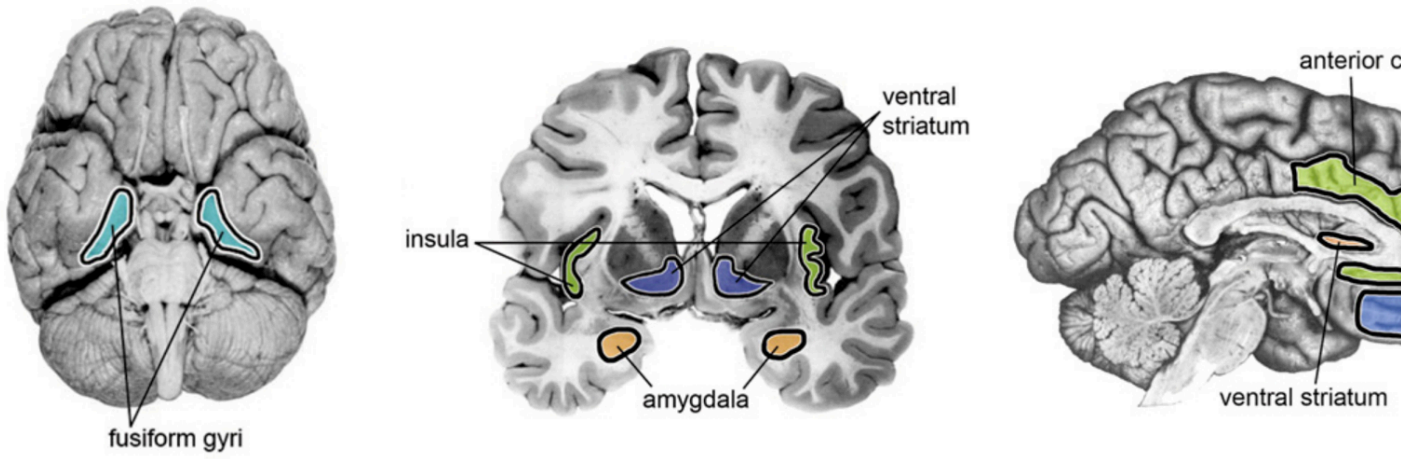


Fig. 1. Anatomical images of several key brain regions associated with group categorization and evaluation, functional relations between groups, empathy, and prosocial and antisocial behavior. As such, this figure is meant to serve as a guide to the location of various regions we reference frequently (not to represent a neural circuit supporting one process in particular). mPFC = medial prefrontal cortex; OFC = orbitofrontal cortex.

Figure 1.4: Figure 1 reorganized with original caption from Cikara and Van Bavel (2014)

to the *stochastic block matrix* algorithm for creating structured random graphs (Rombach et al. 2014; Milzman and Moser 2023). Homophily is the measure of how much more likely an individual is to socially connect within their own group versus with a member of a different group. Homophily can range from -1 to +1, where -1 represents no within-group connections and only between-group connections (i.e., *anti-homophily*); 0 represents an equal probability of within- and between-group connections, and +1 represents only within-group connections. We will define homophily as either a global or group-level variable, though homophily could vary individually as well. There are two types of homophily:

- **Choice Homophily**: Individuals actively prefer interacting with similar others.
- **Induced Homophily**: Social interactions limited by historical or external conditions like geography, profession, birthplace, etc.

These structural elements can significantly limit the diffusion of sustainable practices from peripheries, like the mangrove management on smaller islands, to central cores. However, as colleagues and I have showed, this core-periphery structure, defined by moderately high majority-group homophily can actually *promote* the diffusion of adaptations, provided the adaptation is practiced by the minority group (Turner et al. 2023), as is the case for mangrove forest management or prescribed burns. Formal definitions of homophily and core-peripheriness will be given in the networks chapter.

1.5 Course outline in the context of sustainable development

The agent-based modeling approach developed in this course provides a structured way to test the effectiveness of social interventions aimed at promoting sustainable behavior. By formalizing and testing our assumptions about social dynamics, agent-based modeling supports better design and evaluation of policies and programs aimed at driving real-world change. This deductive, experimental approach allows us to explore how cooperation, coordination, identity, homophily, and influence affect the likelihood that beneficial behaviors will spread. In the coming chapters and associated problem sets we will analyze social learning of behaviors and social influence of opinions in various contexts.

The goal is to build up a repertoire of agent-based modeling techniques for incorporating different assumptions about how social learning or influence work, whether group structure is important to these processes, and for modeling social network structure. This repertoire can then be applied to sustainability contexts of interest to understand how different sustainability intervention strategies, such as who should learn about sustainable adaptations first in an educational intervention or how best to assist stakeholder deliberations to reduce opinion polarization that can derail collective adaptation. To choose model components wisely requires an understanding of elements of sustainability, cognitive and social science, network science, and software engineering.

The UN Sustainable Development Goals help us focus and organize our work by providing concrete goals for evaluating progress towards sustainable development for all Figure 1.5. These goals include targets for institutional development that promotes basic conditions for human thriving (justice, equality, education, public health, and no poverty) so as to assemble and enable a critical mass of people to participate bringing about sustainable development. People cannot participate in sustainability if they suffer in poverty, from illness, or subjugation by authoritarians–all but the most zealous environmental defenders will fight on when these basic needs are unmet. Since progress has been slower than necessary.

Sustainability, then, has several different dimensions, all of which contribute to climate action and environmental protection. I have organized these goals into a coding system called the vIBE system: ***v****ibrant* ***I****nstitutions* support ***B****asic* human needs of people who protect the ***E****nvironment*. All 17 goals and the organization system are illustrated in Figure 1.5. Organizing and connecting our work to has two benefits. First, it helps us identify which cognitive and social factors are at work in different sustainability foci. Second, it expands the corpus of existing research on which we draw to consolidate our social and cognitive theories of behavior change that we will apply to sustainability interventions.

Figure 1.5: The 17 United Nations Sustainable Development Goals (SDGs) can be organized into groups for vibrant Institutions (Group I) that provide Basic needs (Group B) required for people to actively work towards Environmental and Ecological protection (Group E)–we dub this the vIBE coding of the sustainable development goals.

1.6 Computational social science roots

One goal and theme of `socmod` is to avoid unneccesary theoretical commitments to either **epidemiological forcing**-style assumptions (e.g., using SI/SIS/SIR) or to **population genetics** (e.g., Kimura–Crow models) analyses that represent social learning as evolutionary selection. The **agent-based social learning** framework provided by socmod accommodates these or other various social behavior models. In the case of diffusion of adaptations, these formalisms share a common mathematical structure of **birth–death processes**.

We compare these modeling approaches in two ways: first, through their fixation dynamics under weak selection or forcing, and second, through how they each map to a birth–death framework. This is meant to situate users who think in either formalism to how they can map their domain knowledge onto socmod's framework for agent-based models of social behavior.

1.6.1 Contagion, selection, and social learning

Fixation dynamics arise when a type, trait, or state spreads through a finite population until it completely replaces alternatives. Across epidemiology, population genetics, and social learning, this process can often be formalized using **birth–death models**, where individuals probabilistically transition between types.

The general structure is:

$$\frac{dp}{dt} = \text{birth rate} \times (1 - p) - \text{death rate} \times p$$

where $p(t)$ is the frequency of the type of interest at time t.

- **Birth rate** describes how often new instances of the type arise (e.g., infections, successful adoptions, reproductions).
- **Death rate** describes how often the type is lost (e.g., recoveries, abandonment, replacement by competitors).

In biological contexts, these are referred to as **birth–death processes**. However, for social learning of adaptive sustainable behaviors, a more accurate description would be **learning–unlearning processes**: individuals can adopt and later abandon behaviors depending on social influences, network structure, and environmental feedback.

Belief dynamics, which govern the spread and change of *internal* mental states rather than *observable* behaviors, differ substantially in mathematical structure. Nonetheless, belief dynamics can also exhibit **asymptotic, closed-form approximations** under appropriate assumptions.

We focus first on behavioral diffusion processes grounded in social learning. Belief dynamics and their distinct formal properties will be introduced and analyzed in the second part of this book.

In this section, we compare how fixation arises under weak selection or weak forcing in three domains: forcing models from epidemiology, population genetics models, and agent-based social learning models.

1.6.2 Formal Models: Forcing, Genetic Drift, and Success Bias

In epidemiology, the spread of an infectious disease through a population can be captured by systems of ordinary differential equations (ODEs). For a simple SIS (Susceptible–Infected–Susceptible) model, the equations are:

$$\frac{dS}{dt} = -\beta SI + \gamma I$$
$$\frac{dI}{dt} = \beta SI - \gamma I$$

where: - $S(t)$ and $I(t)$ are the proportions of susceptible and infected individuals at time t, - β is the transmission rate, - γ is the recovery rate.

Fixation of the infection (i.e., the infected state reaching $I = 1$) occurs when transmission outpaces recovery, roughly when $\beta S > \gamma$. In small populations, stochastic drift also matters, especially when I is small.

In population genetics, the probability of fixation of a new mutant allele can be approximated using Kimura and Crow's theory. In the simplest form under weak selection:

$$\text{Fixation probability} \approx p_0 + s(1 - p_0)$$

where: - p_0 is the initial frequency of the mutant allele, - s is the selection coefficient (small positive or negative).

In neutral drift ($s = 0$), the fixation probability equals p_0. When selection is weak but nonzero, the probability is shifted slightly toward the fitter allele.

In social learning models, especially those employing **success-biased learning**, the analogy to selection is direct. Agents are more likely to copy behaviors that have produced higher payoffs or success in the recent past. This introduces a **selection-like bias** in the replication of behaviors:

- **Drift** corresponds to random copying without regard to success.
- **Selection** corresponds to biased copying favoring higher-payoff behaviors.

Thus, success-biased social learning is mathematically analogous to weak selection in evolutionary models: both introduce a systematic force that, over time, increases the frequency of advantageous types.

1.6.3 Fixation with Weak Selection or Forcing

Concept	Forcing Models (Epidemiology)	Population Genetics (Kimura–Crow)	Social Learning (socmod)
Entities	Susceptible/infected states	Alleles at genetic loci	Behaviors or traits in agents
Dynamics	Infection spread vs. recovery	Reproduction with drift and weak selection	Social learning with drift and weak biases
Fixation driver	Transmission β overcoming recovery γ	Drift plus selection (fitness differences)	Drift plus learning biases (e.g., success or frequency bias)
Fixation probability (neutral)	Equals initial infection fraction	Equals initial allele frequency p_0	Equals initial behavior frequency
Fixation probability (with weak selection)	Infection fixates if $\beta > \gamma$	Approx. $p_0 + s(1 - p_0)$ for small selection s	Behavior slightly favored by payoff advantage
Time to fixation	Rapid if $\beta \gg \gamma$; stochastic early otherwise	Slow if drift dominates; scales with N	Faster with strong bias; scales with network size
Role of stochasticity	Important early, less so later	Dominant under weak selection	Dominant under weak learning biases
Modeling framework	Stochastic/deterministic ODEs (SIS, SIR) (Keeling and Rohani 2008; Anderson and May 1991)	Wright–Fisher, Moran, diffusion approximations (Crow and Kimura 1970; Kimura 1962)	Agent-based simulations with probabilistic rules (Boyd and Richerson 1985; Rogers 2003)

- **Forcing models** focus on infection and recovery dynamics.
- **Population genetics** tracks allele frequency changes via drift and selection.
- **Social learning** models behavioral adoption via social influence and drift.

1.6.4 Birth–Death Process Mapping

System	Birth event	Death event
Forcing Models (epidemiology, SIS/SIR)	Infection (Susceptible → Infected)	Recovery (Infected → Susceptible)
Population Genetics (Kimura–Crow)	Allele reproduction (fitness-dependent)	Allele loss (death or replacement)
Social Learning (socmod)	Behavior adoption (social copying or bias)	Behavior loss or switching

All three systems feature: - **Birth**: growth of a type or trait. - **Death**: loss or replacement. - **Drift and bias**: stochastic fluctuations and directional forces.

1.6.5 socmod framework for many models

socmod models, by tuning parameters such as success bias, frequency bias, transmission, and dropout rates, can replicate the core behaviors of both **population genetics** and **epidemiology**.

Key parallels: - Strong success bias ↔ strong selection. - Random copying ↔ neutral drift. - Contagion learning ↔ stochastic infection.

Thus, socmod models offer a flexible platform bridging epidemiology, evolutionary biology, and social dynamics.

2 Techniques for socmod ABM development

This vignette introduces agent-based modeling in `socmod` starting with basic functional and object-oriented programming in R. It therefore builds on an assumed knowledge of R basics which can be gained by studying, for example, *Hands-On Programming with R* or other basic R tutorial introductions.

To model social behavior it helps to have intuitive software tools grounded in an empirically-motivated, self-consistent scientific theory. `socmod` provides those tools, but to put it to best use it's helpful to know how it works, especially for people used to using R in a data science context. Many of the same tools are transferable, especially declarative/functional programming, as is used and encouraged by the tidy approach in R for Data Science.

To create simulations that *generate* social behavior we have to go beyond tabular representations of the world to more complex representations of people and their interactions that can be measured. These measurements are recorded in a standard tabular format, e.g., CSV, which can then be analyzed using tidy/R for Data Science strategies.

In agent-based modeling, we create software representations of simulated people, i.e. *agents*, or other interacting entities, including the agents' environment. *Object-oriented programming* is the natural choice for software design in this case because it provides a structure for defining custom *objects* like socially interacting agents. An *object* is one bit of data that could be a number or character type, but could also be something more complicated. In object-oriented programming we define custom *classes* that specify various data *fields* and function *methods* for maintaining and modifying the state of objects. The fields and methods of a class therefore are sets of related data and functions to represent things in the world.

To summarize, objects are a way to track and modify the state of different software entities. We can create simulations of real-world systems by defining custom object types called classes whose state and behaviors are modeled on relevant real-world features and behaviors.

In `socmod`, we define custom R objects for agent-based models of social behavior using the `R6Class` method that creates a new object type, i.e., a new *class*. We use tidy-style functional programming where helpful to represent model features and dynamics.

In the remainder of this vignette, I will first review variable assignment and data structures in R including vectors, lists, data.frames, and tibbles. We then review relevant topics in functional programming, then demonstrate how to create custom classes in R with the library R6. This tutorial then closes with a demonstration of a simple four-agent agent-based model of social behavior, kept simple to highlight the functional and object-oriented design patterns in

`socmod`, representing `Agent`, an `AgentBasedModel`, and `Trial`classes defined using the `R6Class` function.

2.1 Data collection types in R

In R, there is a hierarchy of data collection types that are necessary to know about for functional programming, which often involves applying a function across every element of a collection of data. Data collections include vectors, lists, data.frames, and tibbles, which we'll cover here.

2.1.1 Collections of basic data types

In computer science, a *collection* is an abstract data type that organizes and stores instances of other data types. In R, the most basic data types are numbers (e.g., `double` or `integer`) and text, whose data type is `character` in R whether it's a single character or several:

```
print(typeof('c'))
```

```
[1] "character"
```

```
print(typeof("social science is kewl"))
```

```
[1] "character"
```

2.1.2 Vectors

R vectors are defined using the `c()` function, e.g., `vec <- c(0, 0, 1)`. The data type of all vector elements must be the same. This is enforced by R coercing data to different types, for example:

```
print(c(0, 0, 1))
```

```
[1] 0 0 1
```

```
print(c(0, 0, "yo"))
```

```
[1] "0"  "0"  "yo"
```

We index vectors using single square brackets, with R indexing starting from 1:

```
vec <- c(1, 2, 3, 4, 5) # equiv to vec <- 1:5 or seq(1, 5)
print(vec[1])
```

```
[1] 1
```

```
print(vec[length(vec)])
```

```
[1] 5
```

2.1.3 Lists

If one wants to keep elements of different types in a single collection, use the R `list`:

```
l <- list(0, 542, "yo")
print(l)
```

```
[[1]]
[1] 0

[[2]]
[1] 542

[[3]]
[1] "yo"
```

Note the visual representation has changed on printout. We now need to use double-square brackets to index list elements themselves:

```
print(l[[3]])
```

```
[1] "yo"
```

If we leave off one of the square brackets we effectively get a sub-list with just one element:

```
print(l[3])
```

```
[[1]]
[1] "yo"
```

This is useful if you want to create a new sub-list with more than one element:

```
print(l[c(2, 3)])
```

```
[[1]]
[1] 542

[[2]]
[1] "yo"
```

2.1.4 Named lists

Named lists are the primary key-value store in R, just like `dict` in Python. It allows us to label entries of the list and access them using double square brackets with the character name or using the `$` access operator:

```
named_l <- list(a = c(0, 5, 6), b = c(7, 8, 9))
print(named_l$a == named_l[["a"]]) # compares element-by-element
```

```
[1] TRUE TRUE TRUE
```

```
print(all(named_l$b == named_l[["b"]]))
```

```
[1] TRUE
```

2.1.5 `tibble` (and `data.frame`)

Both the `tibble` and `data.frame` classes represent tabular data, meaning data that can be represented in table format, e.g., in comma- or tab-separated value format. There are some subtle differences listed below, but we'll use `tibble` for representing our tables.

Tibbles are like fancy lists that have special properties that make data manipulation more efficient. The details of how this works aren't super important at this point. The important thing to know is how one can initialize tibbles and access tibble columns similarly to lists, as shown in the following example:

```
tbl <- tibble::tibble(a = c(0, 5, 6), b = c(7, 8, 9))
print(tbl$a)
```

```
[1] 0 5 6
```

```
print(tbl[["b"]])
```

```
[1] 7 8 9
```

Using tibbles ensures that data manipulation and analysis using the tidyverse will work as expected. For example, the tidyverse provides functions for analyzing different groups of data within a larger dataset. This is a common data analysis pattern called split-apply-combine, which in the tidyverse translates to group-by and summarise when using the `dplyr` library. For example, we can calculate the mean of measurements in "experimental" and "control" conditions in some fake data:

```
observations <- tibble::tibble(
  condition = c("experimental", "experimental", "control", "control"),
  measurement = c(13.5, 14.6, 3.4, 5.4)
)

mean_measurement_tbl <-
  observations %>%
  dplyr::group_by(condition) %>%
  dplyr::summarise(mean_measurement = mean(measurement))

print(mean_measurement_tbl)
```

```
# A tibble: 2 x 2
  condition    mean_measurement
  <chr>                   <dbl>
1 control                   4.4
2 experimental             14.0
```

2.1.6 Difference between `data.frame` and `tibble`

The `tibble` library provides a table data representation (also called `tibble`) that is a bit more flexible and intuitive than the R built-in `data.frame`. One reason I prefer tibbles is because

traditional data.frames automatically convert strings to factors unless you tell them not to. Tibbles don't, so you're never surprised.

Column naming is also more flexible with tibbles. A data.frame requires syntactically valid R names, while tibbles can handle column names that include spaces or even non-standard characters. I often use this feature to use column names like `Mean adaptation success` that print nicely when used as labels in `ggplot`.

Finally, tibble operations always return a tibble. For example, if `df` were a data.frame, the operation `df[, 1]` would return a vector by default. If it were a tibble it would return another tibble, making behavior more predictable in data analysis pipelines.

2.2 Functional programming

Functional programming is especially useful for writing code that applies the same function to several inputs, but wants to leave it up to the user to specify exactly which function should be applied. In `socmod` we pass functions as arguments to other functions to specify how agents pick their interaction partners (i.e., teachers in a learning context) and how social learning works. Here are some simple examples demonstrating the key concepts and techniques of functional programming that could help with `socmod` programming.

2.2.1 Higher-order functions: functions with function arguments

In R, and other programming languages supporting the functional style, one can treat functions like any other data and pass it as an argument to other functions. A function that accepts a function as an argument is called a *higher-order function*. Here is a simple example of a higher order function that takes some `data` as a first argument and applies the function `f` to that data twice, putting the result of each calculation in a two-element vector.

```
repeat_2_higher_order_func <- function(data, f) {
  return (c(f(data), f(data)))
}
```

Below we call this higher order function by providing `data=2` as the first argument and an *anonymous function* as the second argument, which can be written using `\(arg) { ...function body... }` syntax:

```
# \(x) {...} is equivalent to function(x) { ... }; these are
# anonymous functions.
# Expecting to return c(4, 4)
repeat_2_higher_order_func(data = 2, f = \(x) { return (x * 2) })
```

```
[1] 4 4
```

The above is the representation of a vector printed to screen, so we see that our expectations were matched.

```
c(4, 4)
```

```
[1] 4 4
```

2.2.2 `map`: a common, useful higher-order function used often in `socmod`

One of the most useful and common higher-order functions is the map function. This function *maps* a function, denoted `.f` below, onto every element in a collection, denoted `.x`. This notation is from the `map` family of functions in the `purrr` library for tidy functional programming.

```
library(purrr)
```

Here is an example of mapping an anonymous function that multiplies its input by 3 onto a vector with entries 2 and 8:

```
# Now get we use map_vec that applies .f to every element of .x,
# expecting the following to return a vector with elements 3*2
# and 3*8, i.e., c(6, 24).
purrr::map_vec(.x = c(2, 8), .f = \(x) { return(3 * x)})
```

```
[1]  6 24
```

If we want to apply a function that takes two variables, the value and index of an element in a collection, we can use the `imap` family of functions as follows to return a vector that contains the original element multiplied by its place in the input vector:

```
input_vec <- c(2, 5050, 6)
purrr::imap_vec(input_vec, \(el, idx) { return (idx * el) })
```

```
[1]     2 10100    18
```

2.3 Custom objects: R6 classes in socmod

Classes are ways to *encapsulate* diverse distinct, but related, processes, behaviors, data, attributes, and other types of information in a single *object*, i.e., a software representation of an *instance* of that entity. R6 is a library for creating our own custom classes that serve as an abstract template that specifies what distinguishes different types of objects/entities. Below we show first how to create a new agent, i.e., a new instance of the `Agent` class that is provided by `socmod`. After that is a fun example of how we can design a different social behavioral model, one of football/soccer matches. We write classes for players and teams and develop a `play_match(team1, team2)` function that pits two teams against each other.

2.3.1 The `Agent` class in `socmod`

Below we create a new instance of the `Agent` class using the class *constructor*, the function written `socmod::Agent$new()` below that creates a new instance of the class.

```
a1 <- socmod::Agent$new(1, name = "Matt",
                        behavior = "Adaptive",
                        fitness=1e6)
```

We can use the *access operator*, `$` in R, to access the *fields* (i.e., *attributes*) of agent `a1` like so:

```
print(a1$get_fitness())
```

```
[1] 1e+06
```

```
print(a1$get_name())
```

```
[1] "Matt"
```

```
print(a1$get_id())
```

```
[1] 1
```

```
print(a1$get_behavior())
```

```
[1] "Adaptive"
```

We can also use `purrr::map` functions over neighbors like so:

```
# Assign neighbors to a1.

a1$set_neighbors(c(
  socmod::Agent$new(id = 2, name = "n2"),
  socmod::Agent$new(id = 3, name = "n3")
))

# Get the list of neighbors back to check it worked.
neighbors <- a1$get_neighbors()
print(class(neighbors))  # should be [1] "Neighbors" "R6"
```

```
[1] "Neighbors" "R6"
```

```
# Neighbors$map() returns a list...
neighbor_names <- neighbors$map(\(n) n$get_name())
print(neighbor_names)
```

```
[[1]]
[1] "n2"

[[2]]
[1] "n3"
```

```
# ...use `unlist` to convert it to a vector:
print(unlist(neighbor_names))
```

```
[1] "n2" "n3"
```

2.3.2 Exercise: design and define classes from scratch

In class we started creating our model of a soccer player agent called `Footballer`, defined below. We ran out of time at the end of class to write *methods* for `Footballer`, i.e., ways that a football player could interact with the world, or that the world could act upon a soccer player. However we only had time to create a *stub* for two methods. A stub is a minimal chunk of code that does very little to nothing, but doesn't get in the way by causing errors or anything like that. It enables us to document our plans for future development in the exact place where it would happen in the code.

Below we have stubs for `score_goal` and `get_penalty` methods for in-game behaviors. Other possibilities could include `get_traded` that would change its team and perhaps `get_signed` for cases where a player is a free agent.

```
library(R6)

Footballer <- R6Class("Footballer",

  public = list(
    # Listing attributes as fields and
    # setting to zero for their definition.
    speed = 0.0,  # units of max km/h
    accuracy = 0.0, # probability of scoring on a shot
    market_value = 0.0, #
    aggressiveness = 0.0, # units of penalties per match
    team = "",

    initialize = function(speed = 15,
                          accuracy = 0.2, market_value = 1e6,
                          aggressiveness = 0.5,
                          team = "Free agent") {
      self$speed = speed
      self$accuracy = accuracy
      self$market_value = market_value
      self$aggressiveness = aggressiveness
      self$team = team
    },

    # Stub two SoccerPlayer class methods...
    # ...one for scoring a goal in a game...
    scored_goal = function() {
      return (ifelse(runif(1) < self$accuracy, 1, 0))
    },
    # ...and one for getting a penalty in a game.
    get_penalty_on_play = function() {
      return (runif(1) < self$aggressiveness)
    }
  )
)

Team <- R6Class("Team",

  public = list(
```

```
    name = "",
    players = list(),
    wins = 0,
    ties = 0,
    payroll = 0,
    games_played = 0,

    # Team name is required, with players optionally specified.
    initialize = function(name, players = list()) {

      self$name = name

      # Initialize players and payroll.
      for (player in players) {
        self$payroll <- self$payroll + player$market_value
        player$team <- name
      }

      self$players <- players
    },

    # Add a player to the roster.
    sign_player = function(player) {
      # Add the player to the team.
      self$players <- c(players, player)
      # Update payroll.
      self$payroll <- self$payroll + player$market_value
      # Update the player's team to be this team's name.
      player$team <- self$name
    }
  )
)
```

```
footballer <- Footballer$new(accuracy = 0.5)
footballer$scored_goal()
```

```
[1] 1
```

```
# sum(purrr::map_vec(1:10, \(.) p$scored_goal()))
```

We can define a function that simulates playing a football match where the team that scores more goals wins, or a tie if teams have the same score. We track penalties received

but assume they don't have an effect on the match outcome. We assume each player gets ten chances to score a goal. We leave it to the reader as an exercise to make use of the `Footballer$get_penalty_on_play()` method within the game to add penalties and consequences in the game (e.g., penalty kicks or something), and to use the `Team$sign_player()` add more players to each roster, which should increase the mean total scores in each game based on the simple way we've modeled gameplay in `play_match`.

```
# Define a function that models a football match.
play_match <- function(team1, team2) {

  # Walk over each player to see how many scores they get,
  # summing to get the total team score.
  team1_score <- sum(
    purrr::map_vec(
      team1$players,
      \(player) {
      # Calculate the total goals scored by the current player.
        sum(purrr::map_vec(1:10, \(.) player$scored_goal()))
      }
    )
  )

  team2_score <- sum(
    purrr::map_vec(
      team2$players,
      \(player) {
      # Calculate the total goals scored by the current player.
        sum(purrr::map_vec(1:10, \(.) player$scored_goal()))
      }
    )
  )

  if (team1_score == team2_score) {
    team1$ties <- team1$ties + 1
    team2$ties <- team2$ties + 1
    cat("Tie game!\n")
  } else if (team1_score > team2_score) {
    cat(team1$name, "wins!!!\n")
    team1$wins <- team1$wins + 1
  } else {
    cat(team2$name, "wins!!!\n")
    team2$wins <- team2$wins + 1
  }
```

```
  cat(team1$name, ": ", team1_score, "   ",
      team2$name, ": ", team2_score, "\n")

  team1$games_played <- team1$games_played + 1
  team2$games_played <- team2$games_played + 1
}
```

First we need teams, which we create and to which we assign players using the constructor, `Team$new()` in the code below.

```
# Initialize two teams, Whales and Squirrels, each with two
# players. All players have identical default attributes.
whales <- Team$new(
  name = "Whales",
  players = c(Footballer$new(), Footballer$new())
)
squirrels <- Team$new(
  name = "Squirrels",
  players = c(Footballer$new(), Footballer$new())
)

# Play three matches.
play_match(whales, squirrels)
```

```
Whales wins!!!
Whales :  4     Squirrels :  0
```

```
play_match(squirrels, whales)
```

```
Squirrels wins!!!
Squirrels :  5     Whales :  2
```

```
play_match(squirrels, whales)
```

```
Whales wins!!!
Squirrels :  3     Whales :  6
```

```
# Print how many games each time won.
cat("\nAfter three games...", "\nThe Whales have won",
    whales$wins, "games and the Squirrels have won", squirrels$wins)
```

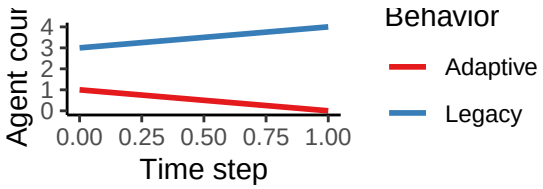
```
After three games...
The Whales have won 2 games and the Squirrels have won 1
```

2.4 Agent-based models of social behavior

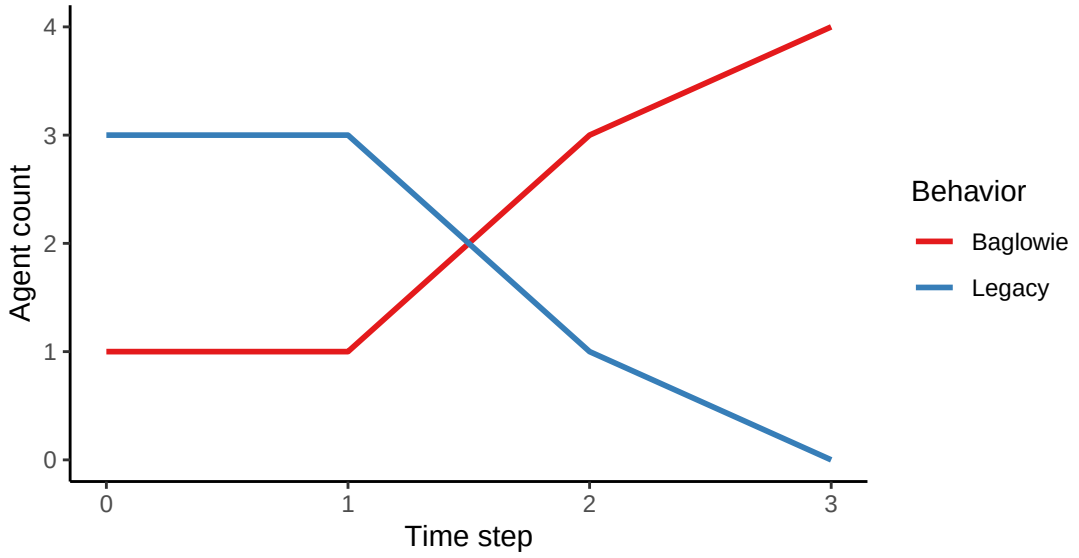
Here we develop our first agent-based models of social behavior. We'll start small, with the following four-household network of solar panel electrification using only success-biased learning. Then we'll do a detailed comparison of different social learning strategies with different parameterizations.

2.4.1 Example 1: Time series of adaptation diffusion

```
adoption_rate = 0.8

# Run model with
g <- igraph::make_graph(~ 1-2, 1-3, 1-4, 3-2)
mps <- make_model_parameters(
  contagion_learning_strategy, graph = g, adoption_rate = 0.8, drop_rate = 0.1
)
cabm <- make_abm(mps);
cabm$get_agent(2)$set_behavior("Adaptive");
trial <- run_trial(cabm, stop = fixated)
plot_adoption(trial, tracked_behaviors = c("Legacy", "Adaptive"))
```



```r
# Same, but now call the adaptive behavior "Baglowie" instead of "Adaptive".
g <- igraph::make_graph(~ 1-2, 1-3, 1-4, 3-2)
mps <- make_model_parameters(
  contagion_learning_strategy, graph = g, adoption_rate = 0.8, drop_rate = 0.1
)

cabm <- make_abm(mps)
cabm$get_agent(2)$set_behavior("Baglowie");

trial <- run_trial(cabm, stop = fixated, adaptive_behavior = "Baglowie")
plot_adoption(trial, tracked_behaviors = c("Legacy", "Baglowie"))
```



2.4.2 Example: comparison of learning strategies and parameters

Agent-based models in `socmod` are composed of sub-models of cognition and social behavior and social networks. One component of agent cognition is its *social learning strategy*. Three social learning strategies are included in `socmod`: success-biased, frequency-biased, and contagion learning strategies see ABM course notes for more details.

Here we show how to compare the three learning strategies across different learning parameters. One parameter we systematically vary affects only success-biased learning, and the other we systematically vary affects only contagion learning. We will design computational experiments to understand their effects, and overall differences between the three learning strategies.

```r
# Load ggplot2 to avoid writing it during plotting below.
library(ggplot2)
# Model generation function used in both computational experiments below.
gen <- function(model_parameter_row) {

  # Extract adaptive_fitness to create agents.
  adaptive_fitness <- model_parameter_row$adaptive_fitness

  agent_1 <- socmod::Agent$new(1, behavior = "Legacy",
                               fitness = 1.0, name = "a1")
  agent_2 <- socmod::Agent$new(2, behavior = "Adaptive",
                               fitness = adaptive_fitness, name = "a2")
  agent_3 <- socmod::Agent$new(3, behavior = "Legacy",
                               fitness = 1.0, name = "a3")
  agent_4 <- socmod::Agent$new(4, behavior = "Legacy",
                               fitness = 1.0, name = "a4")

  agents <- list(agent_1, agent_2, agent_3, agent_4)
  graph <- igraph::make_graph(~ 1-2, 1-3, 1-4, 3-2)

  # Extract other necessary model parameters.
  learning_strategy <- model_parameter_row$learning_strategy
  drop_rate <- model_parameter_row$drop_rate
  adoption_rate <- model_parameter_row$adoption_rate

  # Make ModelParameters to encapsulate this model's parameters.
  model_parameters <- socmod::make_model_parameters(
    learning_strategy, graph, adaptive_fitness = adaptive_fitness,
    adoption_rate = adoption_rate, drop_rate = drop_rate
  )

  return (
    socmod::make_abm(
      model_parameters,
      agents = agents
    )
  )
}
```

2.4.2.1 Experiment over adaptive fitness values

```
# Run the adaptive fitness trials if there is not already a variable
# name for it in the current environment. Change n_trials_per_param to
# a small number like 2-20 for development purposes.
if (!("trials_adaptive_fitness" %in% ls(all.names = TRUE))) {
  trials_adaptive_fitness <- socmod::run_trials(
    gen,
    n_trials_per_param = 100,  # change to 2-20 for shorter runs
    stop = socmod::fixated,
    syncfile = "trials-adaptive_fitness.RData",
    # overwrite = TRUE,
    learning_strategy = c(socmod::success_bias_learning_strategy,
                          socmod::frequency_bias_learning_strategy,
                          socmod::contagion_learning_strategy),
    adaptive_fitness = seq(0.8, 2.4, 0.2),
    adoption_rate = 0.6,
    drop_rate = 0.2
  )
}

trials_summary <- socmod::summarise_by_parameters(
  trials_adaptive_fitness, c("learning_strategy", "adaptive_fitness")
)

trials_success_rate <- dplyr::filter(trials_summary, Measure == "success_rate")
```

Now plot...

```
p <- ggplot(trials_success_rate,
              aes(x=adaptive_fitness, y=Value,
                  color=learning_strategy)
) +
  geom_line(linewidth=1.0) + geom_point(size=2.15) +
  ggsci::scale_color_aaas() + ggsci::scale_fill_aaas() +
  xlab("Adaptive fitness") + ylab("Success rate") +
  scale_x_continuous(breaks = sort(
    unique(trials_summary$adaptive_fitness)
  )) +
  theme_classic(base_size=14) +
  ylim(0, 1.0) +
```

```
  guides(color = guide_legend(title = "Learning strategy")) +
  ggtitle("Adoption rate = 0.6, drop rate = 0.2")

ggsave("vignettes/resources/adaptive_fitness_experiment.png", p,
       width = 5, height = 3)
```

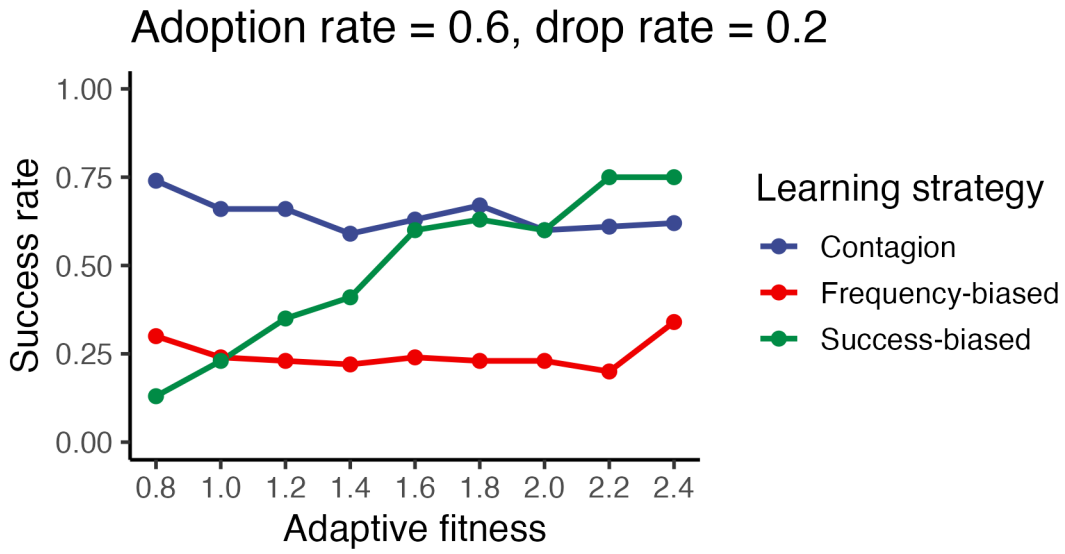


Figure 2.1: Example computational experiment testing effect of adaptive fitness on success rate across the three learning strategies.

2.4.2.2 Experiment over adoption rate values

```
if (!("trials_adoption" %in% ls(all.names = TRUE))) {
  trials_adoption <- socmod::run_trials(
    gen,
    n_trials_per_param = 100,
    stop = socmod::fixated,
    syncfile = "trials-adoption-rate.RData",
    overwrite = TRUE,
    learning_strategy = c(socmod::success_bias_learning_strategy,
                          socmod::frequency_bias_learning_strategy,
                          socmod::contagion_learning_strategy),
    adaptive_fitness = 1.4,
    adoption_rate = c(0.05, 0.2, 0.4, 0.6, 0.8, 1.0),
```

```
    drop_rate = 0.2
  )
}
trials_summary <- socmod::summarise_by_parameters(
  trials_adoption, c("learning_strategy", "adoption_rate")
)

trials_success_rate <- dplyr::filter(
  trials_summary, Measure == "success_rate"
)
```

```
p <- ggplot(trials_success_rate,
            aes(x=adoption_rate, y=Value, color=learning_strategy)) +
  geom_line(linewidth=1.0) + geom_point(size=2.15) +
  ggsci::scale_color_aaas() + ggsci::scale_fill_aaas() +
  xlab("Adoption rate") + ylab("Success rate") +
  scale_x_continuous(breaks = sort(unique(trials_summary$adoption_rate))) +
  theme_classic(base_size=14) +
  ylim(0, 1.0) +
  guides(color = guide_legend(title = "Learning strategy")) +
  ggtitle("Adaptive fitness = 1.4, drop rate = 0.2")

ggsave("vignettes/resources/adoption_rate_experiment.png",
       p, width=5, height=3)
```

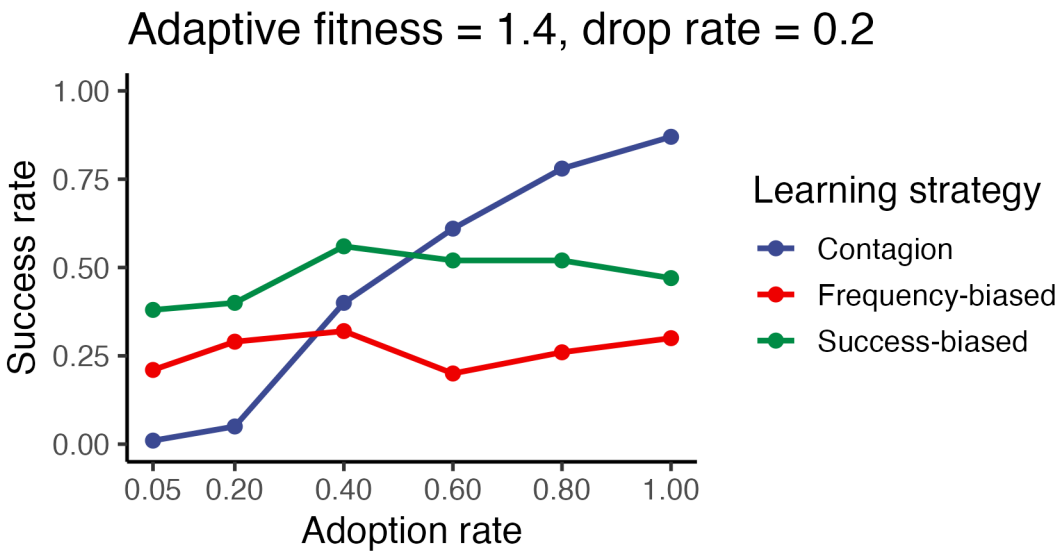


Figure 2.2: Example computational experiment testing effect of adoption rate on success rate across the three learning strategies.

Part I

Diffusion

From whom and how we learn are essential components in explaining the puzzles of diffusion and belief change that underlie the transition to more sustainable practices. In this section we will introduce models of social behavior across different agent-based model configurations. The first chapter in this section introduces a general model of the diffusion of adaptations. Next, we show how models of cooperation can help us understand and predict outcomes in environmental management problems that require stakeholders to do the same behavior, using cooperation with groundwater usage limit laws within a water district as a prototypical example. Next we will present a related behavioral puzzle, that of why certain social norms like gender norms generate inequity and inequality, and point to how modeling social behavior can help us identify potential strategies for reaching more equitable equilibriums.

3 Diffusion

In the context of sustainability and climate adaptation, behavior change often depends not only on individual decisions but also on the social networks through which information and behaviors spread. This chapter examines how the structure of social networks influences the diffusion of adaptations—behaviors or practices that help people adjust to new risks or environmental conditions.

3.1 Primer on interventions (Lecture 3 10 minutes, Lecture 4 review 5 minutes)

- Recall that a *sustainability intervention* is an effort to promote some sustainable behavior or set of behaviors through education, training, or other means.
- In order to simulate interventions, we must develop a sub-model for interventions, starting with classification.
  - Our models of interventions will not be totally disconnected from models of the social systems themselves. Intervention models will specify the expected mode of transmission hypothesized to be most important for the design of the intervention.
  - Intervention models should specify the outcome variable. For now, we will use the simplest version of outcomes included by default in `socmod` agent-based model output.

3.2 Comparing seeding strategies for interventions (Lecture 3, April 7, 2025)

3.2.1 Motivation (10 minutes)

We begin with a concrete example: health behavior interventions in rural Honduran villages. In a study by Aral and Christakis, the researchers used detailed network data to test whether targeting well-connected individuals could increase the spread of health-promoting behaviors. The intervention succeeded not because it directly reached everyone, but because the network structure amplified its effects.

This example also serves as the foundation for an agent-based modeling assignment in which we will explain how *computational experiments* help us understand and predict the effect of different seeding and other intervention strategies.

We will close by contrasting seeding strategies that ignore social structure, as in the (**Airaldi2024?**) work reviewed above, with those that leverage social structure.

3.2.2 Network models and structure (10–25 minutes)

To understand why structure matters, we explore a series of increasingly complex network models. We begin with **deterministic models**:

- **Ring networks**, where each node is connected to a fixed number of neighbors.
- **Complete networks**, where every node is connected to every other node.

These help us isolate basic features like clustering and distance. We then turn to **stochastic models**:

- **Erdős–Rényi random networks**, where links are created with fixed probability.
- **Watts–Strogatz small-world networks**, which combine high local clustering with occasional long-range ties.

The small-world model captures key features of real-world social networks: short average path lengths and local clustering. Examples from the original Watts and Strogatz paper—such as the U.S. power grid and the C. elegans neural network—illustrate how widespread these features are.

An unintuitive feature of many social networks is the **friendship paradox**: on average, your friends have more friends than you do. This emerges because well-connected individuals are more likely to appear in others' networks. It affects who sees new behaviors early, who is most influential, and how we interpret network data. We include a visualization to illustrate this dynamic and emphasize the importance of precise language.

3.2.3 From structure to simulation

Agent-based models help us represent complexity by formalizing how *cognition* and *social structure* interact. Each component is modeled separately. *Cognition* refers to the processes agents use to evaluate and adopt behaviors—drawing on existing knowledge, experience, and social feedback. *Social structure* refers to constraints on who interacts with whom, shaped by acquaintanceship, geography, institutions, and chance. We model social structure as a *network*: a graph where *nodes* represent individuals and *edges* represent relationships through which information or influence can flow.

We design our simulations to evaluate different intervention strategies based on cognitive factors we deem most important, with social structure that is "good enough" for representing a real-world system. by vary which agents start with a given behavior or belief, a process often called *seeding*. For example, in a diffusion model, we might seed a behavior with the most connected individuals, a randomly selected group, or individuals located in a particular community. These differences allow us to explore how network structure interacts with initial conditions to shape diffusion outcomes—helping us identify strategies that are more robust, equitable, or efficient.

3.3 The general diffusion model

1. **Cognition**: how agents evaluate new behaviors;
2. **Social behavior**: how agents learn from others and adopt behaviors;
3. **Network structure**: who interacts with whom.

These components are modular, allowing us to isolate the effects of structure while holding behavior constant.

MODEL SKETCH HERE OF GENERAL DIFFUSION MODEL

- Behavior choice and learning
- Group and social structure
- Outcome measure: simulated success rate

This flexible framework will carry through the rest of the course. In the next section, we extend this model to explore how homophily, asymmetry, and structural inequality affect adaptation outcomes.

3.4 Network structure patterns that shape and constrain adaptation

3.4.1 The good and the bad of social structure and long-range ties

3.4.2 Homophily: choice and induced (10 minutes)

Homophily is the tendency for similar individuals to associate. We distinguish two forms:

- **Choice homophily**, where individuals actively prefer similar partners;
- **Induced homophily**, where external constraints (e.g., institutions, geography) create segregation even without conscious preference.

Two empirical illustrations:

- An analysis of remote-work-eligible occupations shows clustering by education, income, and race.
- A study of minority students in a mostly white MBA program finds that homophily can foster solidarity and resilience.

![Placeholder: Plot that I created based on that data.]

These examples show that homophily can inhibit or support adaptation, depending on the context.

We also discuss experimental results from Centola (2010, 2011), which show that homophilous networks can outperform random ones in spreading health behaviors—especially when reinforcement and social learning are important.

3.4.3 Asymmetric homophily and diffusion (10 minutes)

In some cases, homophily is **asymmetric**. One group (often the minority) is more exposed to the other than vice versa. In the "Minority Incubator, Majority Reservoir" model, this asymmetry allows minority-originating adaptations to diffuse outward more effectively.

This structural asymmetry isn't necessarily intentional—it can emerge from how connections are formed. Yet it has real implications for diffusion, visibility, and adaptation speed.

3.4.4 Core-periphery structure (10 minutes)

In a **core-periphery** network, a dense core of well-connected individuals is surrounded by a sparsely connected periphery. This structure supports fast diffusion through the core but may limit peripheral influence.

The South Pacific Island case provides an example: core-periphery organization enabled stable adaptation strategies under conditions of uncertainty and risk. These designs may emerge deliberately or organically.

3.4.5 When ties break: cumulative loss (5 minutes)

To illustrate the consequences of broken connections, we turn to Henrich's study of tool diversity in Oceania. When long-range maritime ties between islands broke down, isolated communities lost complex technologies.

This example highlights how network fragmentation can halt or reverse cumulative cultural evolution—a key dynamic in adaptation and innovation.

3.4.6 Modeling implications (5 minutes)

Using the same agent-based model, we simulate diffusion on homophilous and core-periphery networks. Holding cognition and social behavior constant, we find:

- Some structures accelerate diffusion;
- Some amplify inequality in who adopts adaptations;
- Some slow or block spread entirely.

These outcomes underscore the importance of network design in policy, development, and sustainability work.

3.4.7 Conclusion and transition (5 minutes)

Social networks don't just transmit information—they shape who hears what, when, and from whom. Homophily and core-periphery structures are not marginal—they are central to how behaviors spread, how inequality persists, and how sustainability strategies succeed or fail.

In the next chapter and lecture, we extend this framework to include **identity**, **polarization**, and the role of **group norms**. These additional dynamics interact with the structures introduced here—and complicate efforts to promote coordination, learning, or fairness across diverse populations.

4 Cooperation

Coming soon!

5 Coordination

Coming soon!

Part II

Beyond diffusion

Choosing what to do is not just defined by payoffs, social structure and a single social learning strategy, as we modeled it in the previous chapters on diffusion. In this part of the notes we expand the scope of our social behavior theory to include more features: opinions and polarization, uncertainty and its effect on social learning, and a more nuanced cognitive learning model called *reinforcement learning*.

Opinions (aka **beliefs**) shape our expectations about the benefits of different behaviors and our openness to learn from others, which in some contexts depends on how similar interaction partners are to one another.

**Environmental uncertainty can reduce, remove, or even reverse the benefits of social learning if environmental variability is unpredictable. If the environment is unpredictable, socially-learned information could become outdated before it is learned, meaning a previously adaptive or sustainable behavior could have become maladptive or unsustainable with the passing of time. For example, monocrop agriculture that tills the ground every season may be less economical and sustainable than it once was given that climate catastrophes occur more frequently, but with unpredictable timing.

Models that flexibly integrate information to update behavioral outcome expectations that guide behavior are called *reinforcement learning* models. We humans have an extensive repertoire of *genetic* cognitive adaptations for dealing with this sort of uncertainty. We update our expectations behavioral benefits based on the stream of information we experience in the form of learning from others while accounting for discrepencies between their personal contexts and ours (Witt et al. 2024). Humans are no slaves to socially-learned information, of course, so these models get us one step closer to more realistic social learning.

6 Polarization

Coming soon!

7 Uncertainty

Coming soon!

8 Reinforcement learning

Coming soon!

Part III

Manual

Techniques for doing ABM for sustainability.

9 Writing and project development

It is a great feeling to close out one stage of a research project by writing in careful detail what we found, how we found it, and why it matters. But one of the most overwhelming challenges for me as a researcher, and I believe many others, is to write linear, logical, and likable research papers. One secret I've learned only recently is that the "writing" starts at the very beginning as one is developing a new project. This doesn't mean that the ideas, motivations, and plan for research projects don't change as the project goes from idea to finished product. But there must be sufficient justification and logic underlying a project's motivation and implementation plan. For a while I've been familiar with the saying "great writing means great editing", but until recently I didn't understand that that means one must constantly be producing content (code or prose, figures, tables, etc.) and revising, reworking, recombining, and reevaluating our research products.

Good scientific writing contextualizes our research questions and goals by introducing existing problems and solutions in a linear, logical way. The methods, results, and analyses are presented in sufficient detail so the reader can re-produce the model and analysis. Scientific results themselves must be contextualized them in terms of our research questions and explains *why* the model outcomes emerged as they did. Good writing finishes with a review of the findings, the model or methods that produced them, the research questions that motivated the study, the impact of the findings for the research questions and more broadly for social behavior/sustainability, for social behavioral sciences, and even for our general understanding of the universe, as appropriate.

Scientific papers share a structure that we will use as a template so one need not start from scratch, called the IMAD (or IMRaD) structure. These acronyms stand for the major section headings of research papers corresponding to the major logical divisions of the content. IMAD stands for Introduction, Model, Analysis, and Discussion, and IMRaD stands for *Introduction*, *Methods*, *Results*, *a*nd *Discussion*.

More details on all this soon.

10 Programming agent-based model analyses

In this Manual chapter we will mostly ignore the theoretical motivations for doing agent-based modeling to focus on developing the technical skills and techniques required to efficiently develop analyses of agent-based model. This is the *development cycle*, i.e., the process for developing and analyzing agent-based models.

1. **Sketch it out:** Write a brief justification and documentation explaining the model. A simple "box and arrow" diagram or table of model variables can be a great help to stay focused when writing code to implement the model. Specify which learning strategies, social networks, initial conditions, stopping conditions, or anything else that will need to decide before you write the code to implement your model.
2. **First code draft:** Program the model in computer code. Start small. Develop a simplified version of the model first if your idea seems complicated. Use a small population size and inspect whether the model dynamics . Prototype the analyses, too, creating time series plots for a range of one (or maybe two) of the hypothized-most-important, explanatory input model parameters or initial conditions, holding other explanatory variables constant to one or a few spot-test values.
3. **Code review and refactoring:** As you program you will likely need to revise your code just as you would revise prose, writing multiple drafts to get the meaning right and clarify your ideas as much as possible. Professional software developers call this process *refactoring*. As you model the process you're interested in you get new insights about it, which will make you want to change the model from what you started with. This is fine! This is refactoring. At this point it is good to experiment, prototype, and tweak until you are sure the model address the problem of interest. Note that through the practice of modeling, the problem statement also gets sharpened since modeling forces us to specify and thereby clarify our theoretical explanations. Prototype all results by analyzing a relatively small number of trials (e.g., 5-20) for all (or nearly all) the input variable values to be tested in the full analysis.
4. Write a draft results section with the preliminary results

11 High-performance computing

Coming soon!

References

Airoldi, Edoardo M., and Nicholas A. Christakis. 2024. "Induction of social contagion for diverse outcomes in structured experiments in isolated villages." *Science* 384 (6695). https://doi.org/10.1126/science.adi5147.

Alongi, Daniel M. 2002. "Present state and future of the world's mangrove forests." *Environmental Conservation* 29 (3): 331–49. https://doi.org/10.1017/S0376892902000231.

Anderson, R. M., and R. M. May. 1991. *Infectious Diseases of Humans: Dynamics and Control*. Oxford University Press.

Boyd, R., and P. J. Richerson. 1985. *Culture and the Evolutionary Process*. University of Chicago Press.

Cikara, Mina, and Jay J. Van Bavel. 2014. "The Neuroscience of Intergroup Relations: An Integrative Review." *Perspectives on Psychological Science* 9 (3): 245–74. https://doi.org/10.1177/1745691614527464.

Crow, J. F., and M. Kimura. 1970. *An Introduction to Population Genetics Theory*. Harper; Row.

Deffner, Dominik, Natalia Fedorova, Jeffrey Andrews, and Richard McElreath. 2024. "Bridging theory and data: A computational workflow for cultural evolution." *Proceedings of the National Academy of Sciences of the United States of America* 121 (48): 1–11. https://doi.org/10.1073/pnas.2322887121.

Ehret, Sönke, Sara M. Constantino, Elke U. Weber, Charles Efferson, and Sonja Vogt. 2022. "Group identities can undermine social tipping after intervention." *Nature Human Behaviour*. https://doi.org/10.1038/s41562-022-01440-5.

Eisenberg, Cristina, Christopher L. Anderson, Adam Collingwood, Robert Sissons, Christopher J. Dunn, Garrett W. Meigs, Dave E. Hibbs, et al. 2019. "Out of the Ashes: Ecological Resilience to Extreme Wildfire, Prescribed Burns, and Indigenous Burning in Ecosystems." *Frontiers in Ecology and Evolution* 7 (November): 1–12. https://doi.org/10.3389/fevo.2019.00436.

Henrich, Joseph. 2015. *The secret of our success*. Princeton University Press.

Jones, James Holland, Elspeth Ready, and Anne C. Pisor. 2021. "Want climate-change adaptation? Evolutionary theory can help." *American Journal of Human Biology* 33 (4): 1–17. https://doi.org/10.1002/ajhb.23539.

Keeling, M. J., and P. Rohani. 2008. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press.

Kimura, M. 1962. "On the Probability of Fixation of Mutant Genes in a Population." *Genetics* 47 (6): 713–19.

Kolden, Crystal A. 2019. "We're not doing enough prescribed fire in the western united states to mitigate wildfire risk." *Fire* 2 (2): 1–10. https://doi.org/10.3390/fire2020030.

Malekpour, Shirin, Cameron Allen, Ambuj Sagar, Imme Scholz, Åsa Persson, Jaime J. Miranda, Therese Bennich, et al. 2023. "What scientists need to do." *Nature* 621 (September): 250–54.

McNamara, Karen E., Rachel Clissold, Ross Westoby, Annah E. Piggott-McKellar, Roselyn Kumar, Tahlia Clarke, Frances Namoumou, et al. 2020. "An assessment of community-based adaptation initiatives in the Pacific Islands." *Nature Climate Change* 10 (7): 628–39. https://doi.org/10.1038/s41558-020-0813-1.

Milzman, Jesse, and Cody Moser. 2023. "Decentralized core-periphery structure in social networks accelerates cultural innovation in agent-based model." In *Proc. Ofthe 22nd International Conference on Autonomous Agents and Multiagent Sys- Tems (AAMAS 2023)*. http://arxiv.org/abs/2302.12121.

Nalau, Johanna, Susanne Becken, Johanna Schliephack, Meg Parsons, Cilla Brown, and Brendan Mackey. 2018. "The role of indigenous and traditional knowledge in ecosystem-based adaptation: A review of the literature and case studies from the Pacific Islands." *Weather, Climate, and Society* 10 (4): 851–65. https://doi.org/10.1175/WCAS-D-18-0032.1.

Pearson, Jasmine, Karen E. McNamara, and Patrick D. Nunn. 2020. *iTaukei Ways of Knowing and Managing Mangroves for Ecosystem-Based Adaptation*. Springer International Publishing. https://doi.org/10.1007/978-3-030-40552-6_6.

Piggott-McKellar, Annah E, Patrick D Nunn, Karen E McNamara, and Seci T Sekinini. 2020. "Dam(n) Seawalls: A Case of Climate Change Maladaptation in Fiji." In *Managing Climate Change Adaptation in the Pacific Region*, edited by W. Leal Filho, 69–84. Springer. https://doi.org/10.1007/978-3-030-40552-6.

Ready, Elspeth, and Eleanor A. Power. 2018. "Why wage earners hunt: Food sharing, social structure, and influence in an arctic mixed economy." *Current Anthropology* 59 (1): 74–97. https://doi.org/10.1086/696018.

Rogers, E. M. 2003. *Diffusion of Innovations*. 5th ed. Free Press.

Rombach, M. Puck, Mason A. Porter, James H. Fowler, and Peter J. Mucha. 2014. "Core-periphery structure in networks." *SIAM Journal on Applied Mathematics* 74 (1): 167–90. https://doi.org/10.1137/120881683.

Smaldino, Paul E, Marco A Janssen, Vicken Hillis, Jenna Bednar, Paul E Smaldino, Marco A Janssen, Vicken Hillis, and Jenna Bednar. 2017. "Adoption as a social marker: Innovation diffusion with outgroup aversion." *The Journal of Mathematical Sociology* 41 (1): 26–45. https://doi.org/10.1080/0022250X.2016.1250083.

Swette, Briana, Lynn Huntsinger, and Eric F. Lambin. 2023. "Collaboration in a polarized context: lessons from public forest governance in the American West." *Ecology and Society* 28 (1). https://doi.org/10.5751/ES-13958-280129.

Turner, Matthew A., Alyson L. Singleton, Mallory J. Harris, Ian Harryman, Cesar Augusto Lopez, Ronan Forde Arthur, Caroline Muraida, and James Holland Jones. 2023. "Minority-group incubators and majority-group reservoirs support the diffusion of climate change adaptations." *Philosophical Transactions of the Royal Society B: Biological Sciences* 378 (1889). https://doi.org/10.1098/rstb.2022.0401.

United Nations. 2023. "Times of crisis, times of change: Science for accelerating transformations to sustainable development." https://sdgs.un.org/sites/default/files/2023-09/FINAL GSDR 2023-Digital -110923_1.pdf.

Van Bavel, Jay J, Dominic J Packer, and William A Cunningham. 2008. "The Neural Substrates of In-Group Bias." *Psychological Science* 19 (11): 1131–39.

Witt, Alexandra, Wataru Toyokawa, Kevin N. Lala, Wolfgang Gaissmaier, and Charley M. Wu. 2024. "Humans flexibly integrate social information despite interindividual differences in reward." *Proceedings of the National Academy of Sciences* 121 (39). https://doi.org/10.1073/pnas.2404928121/-/DCSupplemental.Published.