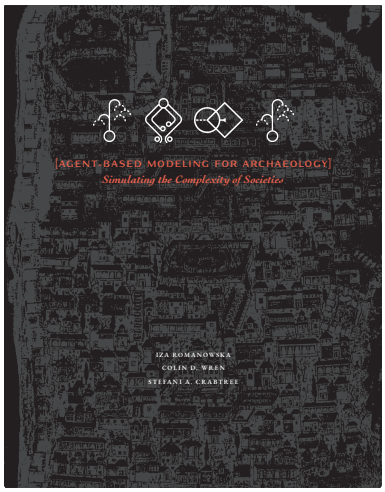
PLEASE NOTE:

The contents of this open-access PDF are excerpted from the following textbook, which is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License:

Romanowska, I., C. D. Wren, and S. A. Crabtree. 2021. *Agent-Based Modeling for Archaeology: Simulating the Complexity of Societies.* Santa Fe, NM: SFI Press.

This and other components, as well as a complete electronic copy of the book, can be freely downloaded at https://santafeinstitute.github.io/ABMA

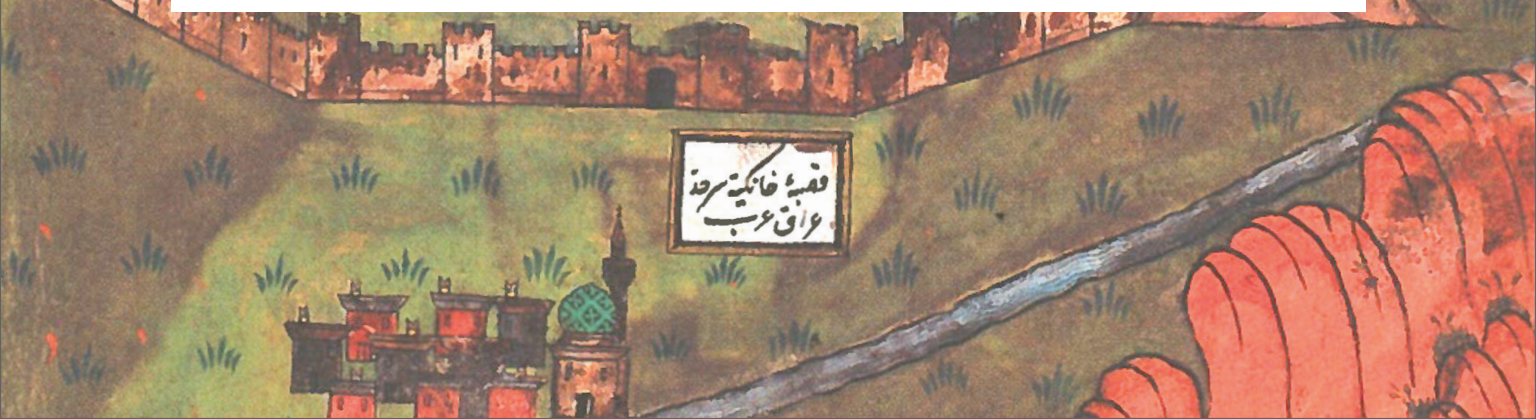



REGARDING COLOR:

The color figures in this open-access version of *Agent-Based Modeling for Archaeology* have been adapted to improve the accessibility of the book for readers with different types of color-blindness. This often results in more complex color-related aspects of the code than are outlined within the code blocks of the chapters. As such, the colors that appear on your screen will differ from those of our included figures. See the "Making Colorblind-Friendly ABMs" section of the Appendix to learn more about improving model accessibility.

SFI PRESS

THE SANTA FE INSTITUTE PRESS 1399 Hyde Park Road, Santa Fe, New Mexico 87501 | sfipress@santafe.edu



# EXCHANGE ALGORITHMS: HOW DO PEOPLE TRADE IN GOODS, IDEAS, AND PATHOGENS?

## 5.0 Introduction

**OVERVIEW**

- ▷ Algorithm zoo: economic interactions and cultural evolution
- ▷ Supply and demand, price setting, barter algorithms
- ▷ Cultural transmission, biases, innovation, and cumulative cultural algorithms
- ▷ Epidemiological modeling
- ▷ Developing model ontologies
- ▷ The rule of parsimony

By now it should be clear that there are multiple ways to represent a phenomenon. Choosing among these representations is a fundamental decision in every modeling study. Which theoretical framework or particular algorithmic representation is most relevant depends on the model's context, the research questions, and, ultimately, the modeler's individual judgment. This chapter will follow the same algorithmic zoo structure as chapter 4, including a full bibliography for the models with links to original code at the end of the chapter. See our book's repository for NetLogo versions of the models.[1]

Compared with movement, models of exchange are usually more complicated, since they involve at least two agents with potentially conflicting goals and strategies. There are whole university departments devoted to studying different models of economic, social, and cultural exchange, and it would require multiple textbooks to discuss them all. Here, we will focus on the most fundamental ones as they often serve as the stepping-stones for more complex models. The model zoo list at the end of the chapter can be used to navigate toward alternative frameworks and more complex implementations.

There are two major families of exchange models in archaeology and related disciplines: one focused on the economic exchange of goods or services, and the other related to the transmission of information. We will look at them separately, but it is worth remembering that they share many elements. In fact, closer integration between these two families of

[1] You can find all code written in this chapter in the ABMA Code Repo: https://github.com/SantaFeInstitute/ABMA/tree/master/ch5

models is a promising direction that still needs more research (Carrignon, Brughmans, and Romanowska 2020).

The classic textbook for ABM in economics is Hamill and Gilbert (2016), while the best introductory texts for cultural evolution are Mesoudi (2011) and Acerbi, Mesoudi, and Smolla (2020).

We will start with the most basic model of exchange, and then expand into the topics of supply and demand, price setting, etc. before moving on to models of cultural transmission. We will also look in more detail at ontology building and parsimony—that is, deciding what goes into and what is left out of our models. This chapter is a stepping-stone for chapter 8, where we will delve into overarching theories of relationship representation, building an exchange model up from a theoretical platform, and then analyzing the output using techniques from network science.

## 5.1 Modeling Commercial Exchange

**Model:**
*Simple Economy* by Wilensky

**ABMA Code Repo:**
*ch5_simple_barter*

To start off on the right foot, it is important to know the difference between trade and exchange. While most (if not all) human groups use exchange as a means of redistributing goods, here we adopt a definition of trade as a type of commercial-based exchange for monetary value. This is obviously a gross simplification, given the many facets of trading interactions, but starting from the most simple representation is the *modus operandi* of modeling.

For other types of economic exchange, see chapter 8.

### BASIC EXCHANGE

You may remember from chapter 2 that in the most basic model of exchange, one agent simply decreases their number of goods while another agent increases theirs.

CODE BLOCK 5.0

```
turtles-own [goods]
to setup
  crt 500 [
    set goods 100
  ]
end
```

Agent location is irrelevant here, so we have ignored it. Agents will be stacked on one cell and won't *appear* to do anything on the map.

CODE BLOCK 5.0 (cont.)

```
to go
  ask turtles with [ goods > 0] [
    set goods goods - 1
    ask one-of other turtles [
      set goods goods + 1
    ]
  ]
end
```

Interestingly, even from such a simplistic model we can draw surprising conclusions. If you run the model for long enough, you will notice that the wealth starts to get accumulated by an increasingly small group of agents. To see it clearly, use a histogram plot with a PEN UPDATE command: `histogram [ goods ] of turtles`. To format the plot as a histogram, click on the pen and change the MODE drop-down menu to BAR and the interval to 10. You will need to adjust the $x$- and $y$-axis ranges as well to suit the data (use `set-plot-y-range 0 40` in the plot command box).

You can also use a MONITOR with `max [goods] of turtles` in the REPORTER box to see the "wealth" amassed by the richest turtle.

After some time, a few agents will hold the majority of goods while the rest have very little, even though the exchange mechanism is entirely random (fig. 5.0). This unexpected pattern is a reminder that even a seemingly basic model, simplified to such an extent that it holds little resemblance to the real world, may surprise us. We know that people do not present strangers with goods at random. However, having seen that inequality can emerge from simple random interaction, this should give you pause when you approach simulations involving more complex dynamics.

Even when checked, the AUTO-SCALE? box will not automatically update the $x$-axis range for a histogram.

**TIP**

You can adjust the code so that turtles can have negative goods (i.e., be in debt). Notice how it affects the wealth of the richest turtles.

A natural extension of this simple model would be the introduction of a second type of good so that agents engage in barter. To do so, simply create two new turtle variables `goodsA` and `goodsB` with randomly assigned amounts (i.e., `set goodsA random 100`) and let the agents exchange one for the other.

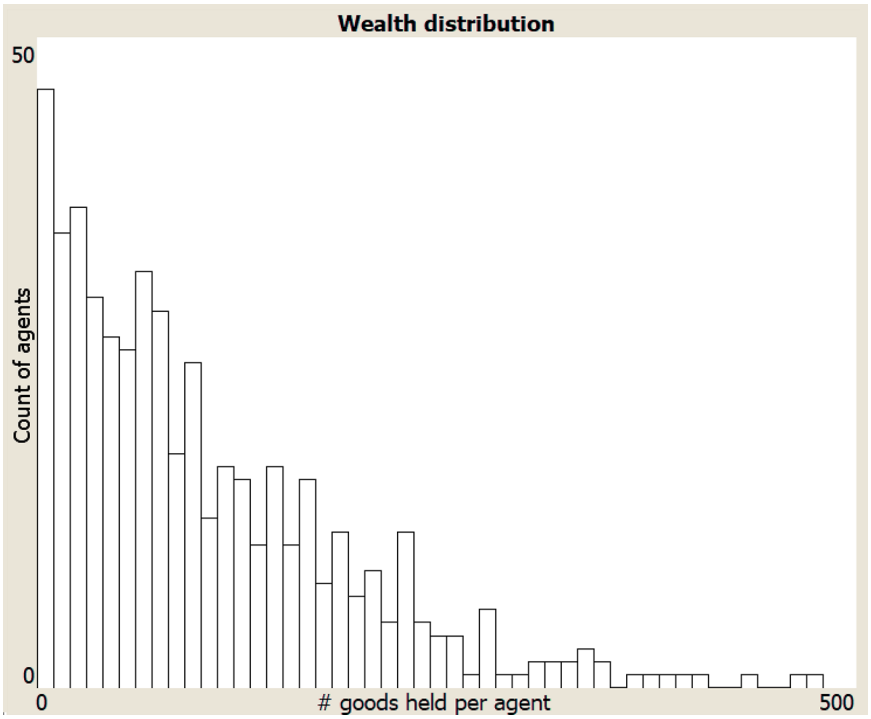



**Figure 5.0.** *Simple Economy* model. Histogram shows a long right tail to the wealth distribution.

CODE BLOCK 5.1

You can use separate histograms to visualize the distribution of each good.

```
ask turtles [
  ifelse goodsA > goodsB [
    let sellers turtles with [ goodsB > goodsA and
    goodsB >= 0 ]
    if any? sellers [
      ask one-of sellers [
         set goodsB goodsB - 1
         set goodsA goodsA + 1]
      set goodsB goodsB + 1
      set goodsA goodsA - 1
    ]
  ][
```

CODE BLOCK 5.1 (cont.)

```
      let sellers turtles with [ goodsA > goodsB and
      goodsA >= 0]
      if any? sellers [
        ask one-of sellers [
          set goodsA goodsA - 1
          set goodsB goodsB + 1
        ]
        set goodsA goodsA + 1
        set goodsB goodsB - 1
      ]
    ]
  ]
```

Here, we asked agents to aim to have an equal amount of each good, but this is just the simplest baseline. In the real life it would be rare for the utility of one item to be exactly equal to another item, which brings us directly to the laws of supply and demand.

### SUPPLY & DEMAND

Null models are often just one step simpler than you think is necessary to model your system. They provide a baseline output for comparison to the actual model dynamics you want to evaluate.

**Model:**
*Market* by Hamill and Gilbert

**ABMA Code Repo:**
*ch5_supply_demand*

The most fundamental extension of the null model above is the notion that humans sell and buy material goods depending on their needs, thus their trading is governed by the laws of supply and demand. A general rule of capital is that although buyers want to buy and sellers want to sell, what is in demand may not always match what is available, and not everyone will go home satisfied. In the first chapter of Hamill and Gilbert's 2016 textbook in agent-based economics,[2] the authors show a simple model of a market where agents can buy goods they need, such as fruit. We recreate it here very closely but with a small archaeological twist of trading pottery.

The model consists of a few traders with stocks of various pot types, and more numerous buyers with individual shopping lists. Both are drawn randomly from a global list of pottery types: `goods`. Not every

[2]You can download the book, courtesy of the authors: https://hamill.co.uk/lynne-hamill/abm-in-economics.

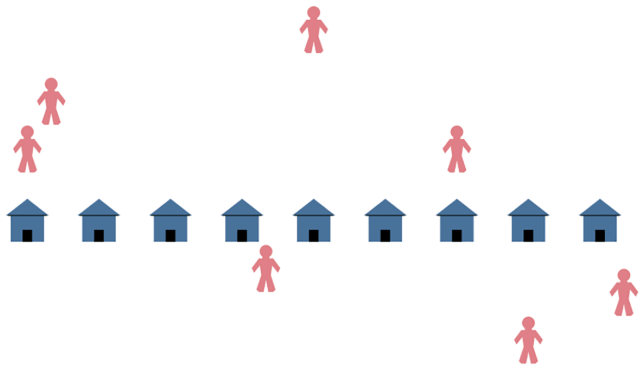


**Figure 5.1.** Screenshot of the original *Market* model. Shoppers choose products available at different traders represented as houses.

seller will have all items a given buyer needs, so buyers will have to shop around (fig. 5.1).

CODE BLOCK 5.2

```
globals [goods]

breed [ buyers buyer ]
buyers-own [ shopping-list my-home ]

breed [ sellers seller ]
sellers-own [ stock ]
```

This is a classic example where using two breeds of turtles (sellers and buyers) makes coding much easier. Each breed has a distinctive set of behaviors and variables, but they also have some in common, for example, `goods`. We usually differentiate them with color and shape (fig. 5.2).

CODE BLOCK 5.3

When you see unfamiliar primitives in the following code blocks, check the NetLogo Dictionary.

```
to setup
  ca
  set goods [ "potA" "potB" "potC" "potD" "potE"
  "potF" "potG" ]
```

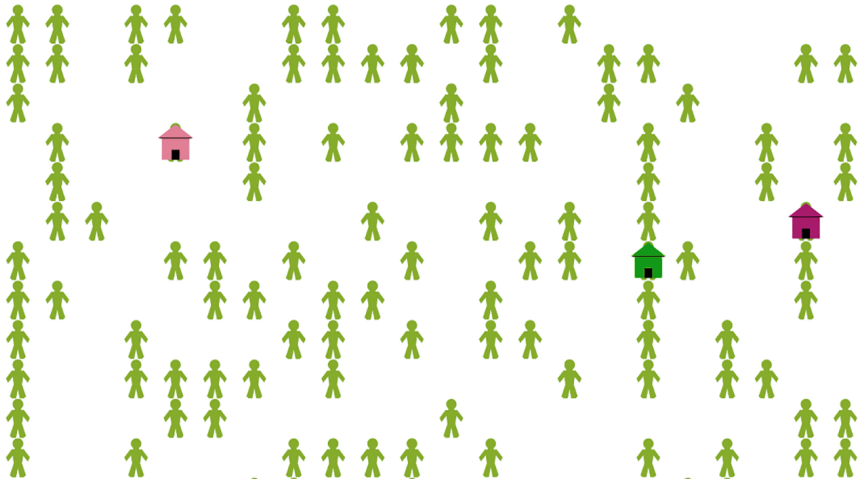


**Figure 5.2.** Screenshot of the modified *Market* model. As in the original model, shoppers are represented as people and traders as houses, but we have many more shoppers, and traders can be differentiated by their color.

CODE BLOCK 5.3 (cont.)

```
  ask n-of 7 patches with [count turtles-here = 0] [
    sprout-sellers 1 [
      set shape "house"
      set stock n-of 3 goods
    ]]
  ask n-of floor ( count patches * 0.5 )  patches with
  [count turtles-here = 0] [
    sprout-buyers 1 [
      set shape "person"
      setxy random-pxcor random-pycor
      set shopping-list n-of 5 goods
      set my-home patch-here
    ]]
  reset-ticks
end
```

Now we will ask buyers to walk from one seller to another until they find all the items on their shopping list. Once that is done, they can go back home, and we remove that agent from the simulation.

CODE BLOCK 5.4

```
to go
  ask buyers [
    let target one-of sellers
    move-to target
    let purchases filter [i -> member?
    i shopping-list] [stock] of target
    foreach purchases [i -> set shopping-list remove i
    shopping-list]

    if length shopping-list = 0 [
      move-to my-home
      die
    ]
  ]
  if count buyers = 0 [stop]
  tick
end
```

The notation `->` indicates an anonymous procedure, in this case with the temporary variable `i` taking on the values of their shopping lists.

The command `filter` takes two variables, a condition structured as a reporter (using an anonymous procedure here) and a list (the target's stock list), then returns a reduced list with each item for which the condition was true. In this case, the condition was that each item in the `stock` was also a `member?` of the `shopping-list`.

This constitutes the basic building block for a supply–demand-driven model. Here, the supply of sellers is unlimited; buyers know the locations of shops and have unlimited time to do their shopping. Thus, in the end everyone checks off all items from their shopping lists and returns home. Small tweaks, however, will make for a complex model in which some agents will not be able to fulfill their needs.

Archaeologists are more concerned than an average economist with what happens to a purchase after the transaction, so we cannot entirely depend on ready-made models from the economic or social sciences. For archaeology an item's final destination is important because it is what we uncover as part of the archaeological record. Thus, to be able to compare the simulation output with the archaeological record, we need to devise

Obviously, people in the past did not store broken pots in their cellars. It's just a placeholder name. Be aware, though, that "controversial" variable names can occasionally draw critique even if they don't change the functionality of the model.

additional functionality. Fortunately, just a few changes to the supply–demand model will transform it from abstract to archaeologically useful. First, add a "cellar" to each house (i.e., their currently occupied patch) in which agents deposit the goods they bought. Note that the code block below only shows the lines requiring changes to the code above and their location within procedures.

CODE BLOCK 5.5

```
patches-own [ cellar ]
to setup
  ...
  sprout-buyers 1 [
  ...
    ask my-home [
      set cellar [ ]
    ]
  ...
```

Here, `cellar` is first initialized for all patches with 0, then buyers make their patch's cellar a list. Keep track of your data structures!

Now, if we just let buyers deposit all of their shopping in the cellars, we will get a random distribution of pottery in the cellars (since the shopping lists are randomly generated). Instead, we can limit their shopping time and give them a basket to hold items they managed to buy within the allocated time. First, initialize the basket as a `list` in `setup`.

CODE BLOCK 5.6

```
buyers-own [ shopping-list my-home basket ]
to setup
  ...
  sprout-buyers 1 [
    ...
    set basket [ ]
    ...
]
```

Second, rather than just check goods off the shopping list, add them to the basket (this line should be placed just after `let purchases filter...`).

CODE BLOCK 5.7

```
foreach purchases [i -> set basket fput i basket]
```

Finally, add to the condition under which a buyer goes home, so that the buyer only has three `ticks` to complete the shopping.

CODE BLOCK 5.8

```
if length shopping-list = 0 or remainder ticks 3 = 0 [
```

We can now deposit the goods purchased from the sellers, and go back to shopping with a new shopping list. Replace the `die` line with `deposit-pots` and add a new procedure below:

CODE BLOCK 5.9

```
to deposit-pots
  foreach basket [ i ->
    ask my-home [ set cellar fput i cellar ]
  ]
  set shopping-list n-of 5 goods
  set basket [ ]
end
```

Above, `remainder` divides the two numbers but only returns the remainder, not the result of the division. It is commonly used to execute code every X number of ticks instead of every tick.

The final pattern of pot distribution can be visualized in many ways, but let's assume that we want to see the most common type of pottery in each house.

CODE BLOCK 5.10

```
to show-distribution
  let colors [14 24 44 64 84 104 124]
  ask turtles [ hide-turtle ]
  ask patches with [cellar != 0][
    let most-pottery one-of modes cellar
    let potcolor position most-pottery goods
    set pcolor item potcolor colors
  ]
end
```

Note that we use the primitive `one-of` with `modes cellar`, so that if multiple types of pottery are equally frequent only one will be displayed.

This code block features a common trick used for lists: using the index (`position`) from one list (`let potcolor position most-pottery goods`) to access a specific variable (e.g., the pottery's color) in another list (`set pcolor item potcolor colors`). For example, pottery A is the first in the `goods` list, so reading the index with `position`, we can retrieve the first color from the `colors` list. This works because in NetLogo lists are *ordered*; that is, the order of a list's

Normally, we spell out colors, e.g., "yellow," but when grouped in a list, it is easier to refer to them with numbers.

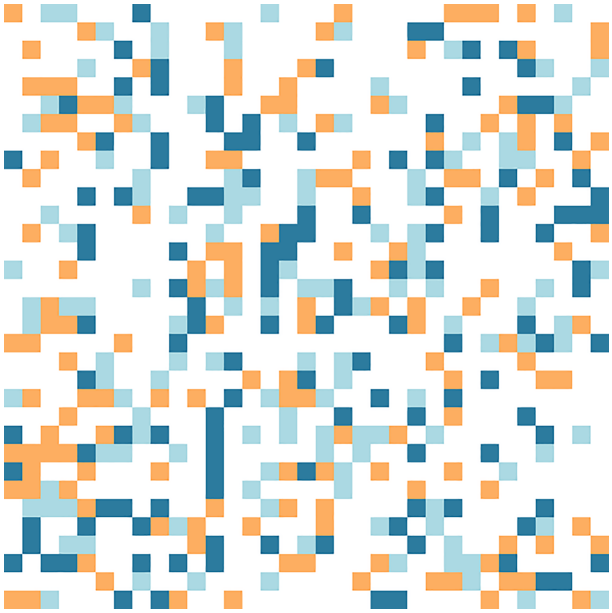


**Figure 5.3.** Model output shows the spatial distribution of the most common pot type in each household.

elements is fixed. Lists are also *immutable*, meaning that they cannot be directly modified even if you can build a new list from an altered copy of the original.

The patterns we simulate will vary depending on whether agents wander at random or know the location of sellers, whether each seller has one unique good to sell or a collection of different types of goods, and whether the number of goods to purchase is unlimited. The resulting map of pottery distribution (fig. 5.3) shows an artificial archaeological record, which gives us an insight as to what kind of exchange processes produce what kind of spatial pattern. This insight can be particularly useful when interpreting spatial distribution of real archaeological material. To make it even more relevant to a particular case study, you might decide to include real topography of the area or known locations of human settlements.

How would you investigate the relationship between the frequency of a given pottery type and the distance to the seller who produced it?

Similarly, the demand of agents can be modeled to a higher level of detail, such as by calculating the utility function for each household depending on its needs. For example, already purchased goods and overall budget can lead to interesting patterns in the resulting distribution of goods.

For a thorough introduction to modeling heterogeneous demand, see Hamill and Gilbert (2016, ch. 3).

Models of economic interactions tend to be developed to a high degree of detail, so in most cases we may simply pick up existing frameworks and adjust them to the needs of our particular case study, research questions, and data. In those situations, it is often tempting to keep some functionality in the code because that is how it was in the original model. However, ultimately every model should stand on its own, that is, contain only these elements (entities, variables, algorithms) that serve its intended purpose. For example, if you use the supply–demand model above but going back home to deposit the purchases is not necessary for your particular research questions, you should remove it from the code.

### PRICE-SETTING MECHANISMS

In more realistic models of commercial exchange, sellers set their prices based on the information about the market they have at a given moment. This information may not necessarily be accurate or always up to date. Agent-based modeling is particularly well suited for this kind of modeling because it facilitates tracking the individual knowledge of each trader. The simplest way for an agent to set a price for their goods is to take an average of the demand and supply, as far as they know. That knowledge may be based only on personal information (local knowledge) or on information gained from other traders (global knowledge). In the *Mercury* model (Brughmans 2015; Brughmans and Poblome 2016), local demand of an agent represents the maximum amount that a trader can satisfy of the total consumer demand for pots. During trading, the agent's demand is decreased by one for every successful transaction. Local supply is modeled as the number of available goods stored from the previous time step or produced by the production centers, and a trader's demand increases by one every time step, up to the maximum value in this time step.

**Model:**
*Mercury* by Brughmans

**ABMA Code Repo:**
*ch5_price_setting*

To calculate the price, agents assess a wide network (fig. 5.4). To determine the global (or at least regional) supply and demand, they ask traders in their network for the values of their local supply and demand. The price is then the average demand divided by the sum of average supply and average demand, including the trader's own knowledge of supply and demand.

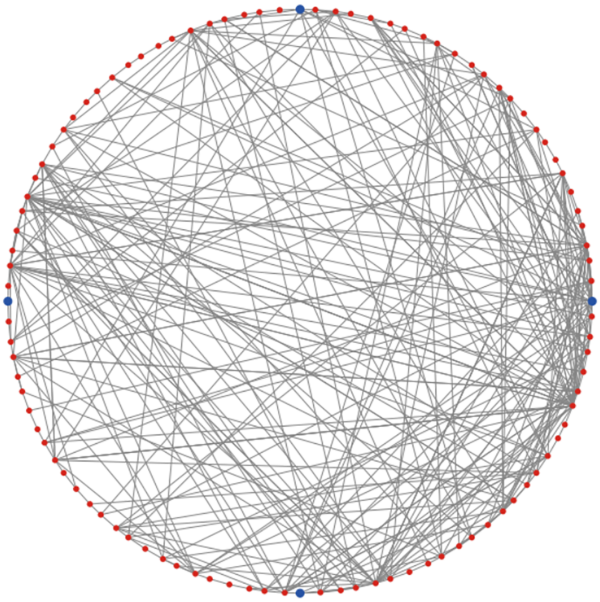


**Figure 5.4.** Screenshot of the *Mercury* model. Trading centers (dots) assess the demand and supply based on information available to them through their commercial network (represented as lines).

```
to-report average-demand
  let sum-demand sum [ demand ] of known-traders
  report ((sum-demand + demand) / ((count
  known-traders) + 1))
end

to-report average-supply
  let sum-prod sum [ products ] of known-traders
  report ((sum-prod + products) / ((count
  known-traders) + 1))
end

to price-setting
  set price (average-demand / (average-supply +
  average-demand))
end
```

CODE BLOCK 5.11

The `+ 1` is added to include the trader who calls the reporter in the calculations.

**Figure 5.5.** Screenshot of the *Edgeworth Box* model. Many economic models have a limited or no spatial aspect. Here patches have been used to show the space of possible outcomes: deal rejected by one or both agents or successful deal (light area).

How to determine which traders are known depends on the type of model. This can be, for example, through spatial proximity—*I know what traders close to me are doing*—or through a network representation—*I know what traders in my network are doing*.

We discuss different types of networks and how to construct them in chapter 8.

This price-setting example is very simple, but we can also follow alternative theoretical frameworks, some of which have been translated into agent-based models by economists. Here we will look at a few examples demonstrating the principles of the process.

**Model:** *Bidding Market* by Baker

In the *Bidding Market* model from the NetLogo Models Library (Baker 2017), the buyers and sellers adjust their asking and buying prices on the basis of the success or failure of previous transactions. That change can be executed using different algorithms:

- random, in which traders randomly change their prices;
- normal, in which both seller and buyer adjust their prices by a small percentage;
- desperate, in which the buyer and seller adjust their prices by a higher percentage; and
- a combination of all three.

For models of generalized market dynamics with price-setting dynamics based on product utility, competition, and profit estimation, see

the models described in Hamill and Gilbert (2016), chapters 5 and 6. In their barter model, agents set the price by individually evaluating the utility of a given product in a barter situation and under different circumstances of supply and demand. The barter model features the basic algorithms for calculating a product's utility for each agent individually (fig. 5.5). Although the model describes a barter situation, it can be easily translated into any exchange mechanism involving money (Hamill and Gilbert 2016, ch. 5). The basic mechanism for establishing the optimal price based on supply and demand is to repeatedly go through a haggling process between two agents.

**Model:**
*Edgeworth Box Game* by Hamill & Gilbert

CODE BLOCK 5.12

```
ask people [ set tempBudget ( goods * priceGoods ) +
money  ]

ask person 0 [
  set optimalGoods round ( tempBudget *
  PersonAPreference / priceGoods )
  if optimalGoods < 1  [ set optimalGoods 1 ]
  set demand  ( optimalGoods - goods )
  if demand < 0 [ set demand 0 ]
  if demand > (  goods  - 1 ) [ set demand ( goods - 1
  ) ]
]
set PersonADemand [ demand ] of person 0
ask person 1 [
  set  optimalGoods round ( tempBudget *
  PersonBPreference / priceGoods )
  if optimalGoods < 1  [ set optimalGoods 1 ]
  set supply  ( goods - optimalGoods )
  if supply < 0 [ set supply 0 ]
  if supply > ( goods - 1 ) [ set supply  ( goods - 1
  ) ]
]
set PersonBSupply [ supply ] of person 1
```

**Model:** *Shops* by Hamill & Gilbert

The authors also provide a basic market dynamics model focused on assessing supply and competition from other producers. Companies can try to estimate the level of production of other companies until they arrive at the **Nash equilibrium**, where none of them needs to change their strategy. The Nash equilibrium is a very useful theoretical concept. In simple terms, it describes a situation in which none of the participants can improve their situation without others changing their strategies—the agents are locked in a specific set of circumstances. This model is too elaborate to present the code in its entirety. However, it is particularly worthy of spending some time on because it translates many economic concepts into ABM language: competition over clients, investment and capacity building, or dynamic adjustment of prices depending on market volatility, etc. (Hamill and Gilbert 2016, ch. 6).

**Nash equilibrium:** a set of agent choices such that no agent can benefit by altering its choice, assuming that the choices of the other agents remain unchanged.

Just like other agent-based modeling topics, there is great heterogeneity in how economic interactions are modeled. In some models, reputation, relatedness, and memory feature heavily in how exchanges are established (Crabtree 2015). In others, specialization in one type of good may be the dominant factor (Cockburn et al. 2013). Finally, some agent-based models rely on almost random interaction, like the model we developed in chapter 2 (Romanowska 2018). From the authors' personal observations in archaeological simulations concerning the deep human past, economic interactions are frequently modeled on a very rudimentary level, often not matching the sophistication of other elements. For example, harvest and soil depletion may be quite realistic, while trade is based on chance interactions alone. On the other hand, models of the Greco-Roman world or other historical periods often represent economic interactions to a much higher degree of detail. This largely stems from the questions researchers are interested in.

For example, Romanists have been discussing the question of the level of economic integration in the Roman Empire for over a century, but archaeologists have only recently used modeling to address it.

The one important way in which classic economic models differ from the ABM approach is how they treat imperfect information and the resulting uncertainty. The standard approach sees economic agents as utility maximizing with full and perfect knowledge (unbounded rationality), perfect self-control (unbounded willpower), and only ever considering their own welfare (unbounded selfishness). While these three assumptions work as null heuristics, they fail to describe particularly well the reality we experience in our daily lives, something behavioral economics and new generations of economists are attempting to address (Farmer

**bounded rationality:** the idea, first proposed by Simon (1956), that humans will often settle on a "good-enough" solution rather than exert time and resources to find the optimal one.

and Foley 2009; Page 2018). Here ABM brings an easy way to loosen these three assumptions of the standard neoclassical model. For example, modeled transactions may be based on an agent's estimates rather than exact knowledge, and traders can behave stochastically using dynamically updated probabilities. Interestingly, in some cases this makes no difference to the results, while in others it does, making this a highly dynamic field of study (Holcombe et al. 2013; Poledna, Miess, and Hommes 2019).

Remember, the ability to represent heterogeneity among individuals is a key advantage of ABM over other modeling approaches.

To draw from the vast collections of existing theoretical frameworks requires a fair bit of effort to learn the foundations of economics. However, once an appropriate model is identified, translating already formalized models into ABM and archaeological case studies is a straightforward task.

Models of trade and commercial exchange focus on the movement of goods and capital. In most cases they consider the utility of each option and the profit or loss for people involved. However, economic interactions do not happen in a social vacuum—they are often embedded in much more complex networks of social interactions involving status, prestige, trust, etc. In the next section, we will look at a family of models that also deals with exchange but of a social nature.

## 5.2 Models of Information Transmission

### VERTICAL & HORIZONTAL TRANSMISSION

Cultural evolutionary theory is a growing field of study concerned with how cultural traits are passed between individuals, groups, and generations and how the evolution of those traits shapes culture. The most thorough recent discussion of cultural evolution by Mesoudi (2011) provides a comprehensive background to understanding the concepts. Cultural evolutionary theory's main framework focuses on cultural transmission (or social transmission), that is, the process of exchanging information between individuals through imitation, teaching, or language-based communication. This process can be vertical (from parents), horizontal (from peers), or oblique (from individuals of a different generation, e.g., teachers), and it occurs in one-to-one (e.g., parent to child) or one-to-many interactions (e.g., teacher to multiple students). Finally, the transmitted values may be blended (e.g., an average of two values) or discrete (e.g., A, B, C). It is crucial to consider whether to model a given cultural trait

**culture:** in this context, any information passed between individuals via social transmission (Mesoudi 2011, p. 2).

through vertical, horizontal, or oblique transmission because each has specific assumptions and will lead to a different rate and characteristic of information diffusion. Some types of information (e.g., related to subsistence) may be predominantly passed vertically from parents to children. Others (e.g., ceramic decoration) may not be as clear-cut—whether we learn them from our family or from peers is far from established.

**Model:** *Cultural Transmission Algorithms* by Romanowska

**ABMA Code Repo:** *ch5_culttrans*

Vertical transmission is the default in NetLogo through the `hatch` primitive.

CODE BLOCK 5.13

```
ask turtles [ hatch 1 ]
```

With `hatch`, each new turtle is an exact copy of the parent if not specified otherwise; thus, any variable representing a cultural trait will be copied exactly as it is. In vertical cultural transmission models, a common simplifying assumption is that the transmission occurs at birth, thus collapsing the child's upbringing into a single event. The `hatch` command takes an optional series of commands for the newly created agent to perform at birth, so we can control their inheritance of cultural traits there. If the child has two parents, you can randomly assign a parent to copy from (e.g., `cult1` below); if the trait is continuous, you can take an average (e.g., `cult2`). Uncomment the second line of code to change the model's behavior from discrete to continuous:

**cultural trait:** different content transmitted between individuals: a skill, reference, piece of information, strategy, or particular behavior.

CODE BLOCK 5.15

CODE BLOCK 5.14

```
let parent1 self
let parent2 one-of turtles in-radius 3
hatch 1 [
  set cult1 [cult1] of one-of ( turtle-set parent1
  parent2 ) ; discrete inheritance
  ;set cult2 mean [cult2] of ( turtle-set parent1
  parent2 ) ; continuous inheritance
]
```

**DON'T FORGET**
Square brackets indicate a reporter, in this case reporting the value of the cultural trait: `cult1` or `cult2`.

Horizontal transmission does not occur at the moment a new agent is created, but rather needs to be triggered by an event during its life. In its most basic form, a **cultural trait** is copied at random from any agent in the population:

CODE BLOCK 5.15

```
set cult1 [cult1] of one-of other turtles
```

This copying event may be triggered by, and limited to, agents in close spatial proximity or related to the agents through other alternative topologies (e.g., within a particular breed or through links in an agent's social network; see ch. 8):

CODE BLOCK 5.16

```
if any? other turtles in-radius 3 [set cult1 [cult1]
of one-of other turtles in-radius 3]
```

If you run this code, you will notice that even in situations where the starting population had 50–50 distribution of a cultural trait and copying is entirely random (known as unbiased transmission), very quickly one trait will take off and outcompete the others. This is a well-described phenomenon known as **random drift** in which some cultural traits become dominant over others as a result of random sampling rather than any inherent advantage they carry (Bentley, Hahn, and Shennan 2004).

**random drift:** random change in the frequency of traits. Often occurs in small populations.

These choices about which mode of transmission to use are part of the model's ontology, which we discuss in detail at the end of this chapter.

### BIASED SOCIAL LEARNING

In most cases though, social learning does not happen at random. What to copy, or who to copy from, is usually a deliberate or culturally mediated decision. We can divide the modes of so-called "biased" social learning into three types following Mesoudi (2011, p. 57): content bias, frequency bias, and payoff bias (fig. 5.6).

#### *Content Bias*

Certain choices are simply better or more attractive than others. For example, some subsistence strategies may produce better caloric returns, or a given cultural trait may be simply more attractive to buyers (e.g., painted pottery vs. plain pottery). Similarly, innovations need to possess certain features that give them a relative advantage to replace existing technologies. Whether the agents are able to correctly assess the difference in utility between different options remains an important consideration in this type of model, but one of the distinguishing features of ABM is that it allows for imperfect information and resulting erroneous choices.

"Unbiased" or "neutral" transmission means "random," like in the previous examples where the value of the trait did not impact its transmissibility.

To model content bias selection, a simple approach is to treat the utility of each cultural trait as a direct representation of its probability of being copied (*the more I like something, the more likely I'll buy/follow/adopt*

In the case of two options that sum to 1, they can be described as optionA and 1−optionA.

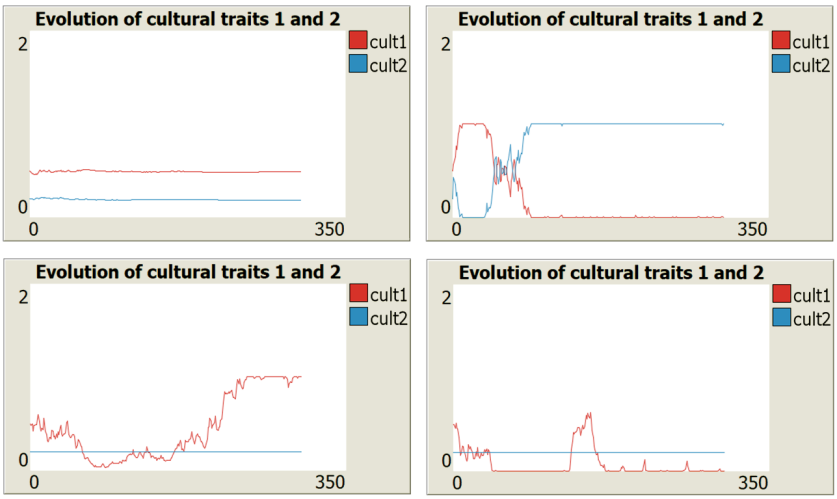



**Figure 5.6.** Different transmission types will result in different trait frequencies. Top, from left: vertical transmission, content bias. Bottom: conformist bias, anticonformist bias. Note that these models include mutation, which is responsible, for example, for the reemergence of the cultural trait in the anticonformist scenario.

*it*). Since the value of each preference is arbitrary, this can be done using a simple list of the probability of options. Here we just use the probability of the first trait, though we could nest `ifelse` statements if there were more traits or if they did not sum to one:

CODE BLOCK 5.17

```
to content-transmission
  let cult_traits (list 0.7 0.3)
  ask turtles [
    ifelse random-float 1 < first cult_traits
    [ set cult1 [cult1] of one-of other turtles ]
    [ set cult2 [cult2] of one-of other turtles ]
  ]
end
```

For longer trait lists with probabilities, look at the ROULETTE WHEEL selection in the `Rnd` extension.

If the traits are mutually exclusive—that is, if you can have either one or the other—then this can be integrated through a set of nested `ifelse` loops. Note that here we are assuming binary trait states such that values are either 0 or 1, rather than using a continuous variable.

CODE BLOCK 5.18

```
to content-transmission-exclusive
  let cult_traits (list 0.7 0.3)
  ask turtles [
    ifelse random-float 1 < first cult_traits [
      set cult1 [cult1] of one-of other turtles
      ifelse cult1 = 1 [set cult2 0][set cult2 1]
    ][
      set cult2 [cult2] of one-of other turtles
      ifelse cult2 = 1 [set cult1 0][set cult1 1]
    ]
  ]
end
```

You may decide to use a content bias algorithm in any context—economic, social, or cultural—in which one option is inherently superior. Depending on what the agents are trying to maximize, this may be due to:

- price (cheaper vs. more expensive);
- quality (better vs. worse);
- personal preference, such as aesthetics, cultural association, or religious beliefs (*I like it* vs. *I don't like it*); and
- novelty and conservatism (new vs. known).

**Model:**
*Cumulative Cultural Evolution Demographic* by Powell et al.

In their work, Powell, Shennan, and Thomas (2009) model transmission leading to accumulating cultural skill in a spatially structured population during the late Pleistocene. They use a combination of vertical and oblique transmission with an important content bias, as their agents choose the most skilled individual to copy from. The availability of teachers to copy from varies depending on model parameters that govern the density of the population and migration. With this model they explain how behaviorally modern cultural traits heralding the onset of Upper Palaeolithic seem to appear and disappear repeatedly in different regions due to the variation in population density and mobility.

### *Frequency Bias: Conformist Bias, Anti-Conformist Bias*

**conformist bias:** probability of adopting a cultural trait proportional to its popularity in the population. The inverse is known as anti-conformist bias.

To study peer influence, researchers often turn to frequency bias. The main premise behind this model is that in certain social situations, people decide on an option based on its popularity. Sometimes popularity is regarded as positive (**conformist bias**), as in *I want what other people have* , and sometimes popularity is regarded as negative (anti-conformist bias), as in *I find this item particularly valuable because no one has it*, for example, luxury goods. Mathematically, these are very easy to express, and their use is almost ubiquitous. As in the content bias example, each cultural trait has an associated probability of being copied; however, rather than being arbitrary, it is expressed as its frequency within the population.

Assuming that 0 signifies the lack of a cultural trait and 1 its presence in each agent's cultural repertoire, its frequency is equal to the mean value of the population. For example, in a population of ten agents, if four of them have the trait, its frequency will be $0.4(0+0+0+0+0+0+1+1+1+1)/10 = 0.4$.

CODE BLOCK 5.19

```
to frequency-transmission-conf
  ask turtles [
    if random-float 1 < mean [cult1] of turtles [
      set cult1 [cult1] of one-of other turtles
    ]
  ]
end
```

The only difference for anti-conformist bias is that it uses the inverse of the frequency:

CODE BLOCK 5.20

```
...
if random-float 1 < 1 - mean [cult1] of turtles
...
```

Conformist and anti-conformist biases are particularly useful for modeling the spread of beliefs and innovations.

Frequency-based bias does not take into consideration any inherent qualities of the selected trait, only its popularity. Thus, it is often combined with content bias. In those cases, the probability of copying is calculated as a function of frequency and utility. Biases of different types can be combined to reflect the complexity of individual decision-making shaping population-level trends.

### *Payoff or Model-Based Bias*

The two previous cultural transmission mechanisms assumed a high level of knowledge (e.g., agents knew the frequency of the trait or its value). This assumption can be easily loosened (e.g., agents may calculate frequency only among closest neighbors), but in some cases the utility of a given trait cannot be directly or reliably measured at all. Instead, individuals need to depend on a proxy to evaluate whether a particular trait is beneficial. These cases are often modeled with a so-called "payoff" or "model-based" bias. An example of this is copying the strategy of the most successful individual (success bias) or individuals with a higher level of prestige (prestige bias). In short, it is not a *trait* that is assessed but the *individual* who carries it. Model-based biases have simple implementations in NetLogo. For example, success bias can be modeled in the following way:

**model-based bias:** probability of adopting a cultural trait based on the characteristics of individuals who carry this trait.

CODE BLOCK 5.21

```
to success-transmission
  ask turtles [
    set cult1 [cult1] of one-of other turtles
    with-max [success]
  ]
end
```

As `success` is a turtle variable, create it in `turtles-own`, initialize the values in `setup`, and update dynamically in `go`.

In this case the variable `success` is dynamically allocated depending on the model. This could be the number of items collected by an agent, the overall profit of its business, or the size of stored resources depending on what is considered a "success" in a given situation.

Prestige bias represents a specific type of success bias in which an individual or a group of agents is considered "special" and their level of success is usually static. This may be as a result of some inherent features (e.g., being a member of the elite) or due to perceived individual qualities (e.g., level of a skill). For example, copying individuals with a higher level of a skill (i.e., teachers) is an established form of learning. Other types of biases will follow the same pattern, replacing the success variable with age (*age bias*), affinity or belonging to the same group/tribe or community (*similarity bias*), etc.

Because we haven't linked success with any particular value of the cultural trait, you'll see how, in different runs, it trends toward either 0 or 1 in the population.

In all of these types of bias, the observed characteristics of the other agent are treated as a proxy (model) for the "quality" of a cultural trait or strategy, and not the trait itself. This creates a framework in which agents

may erroneously copy traits that do not carry an advantage only because a successful individual happened to have them—similar to how people tend to copy the fashion choices of celebrities even though their probability of becoming rich and famous thanks to new shoes is nil.

### MUTATION, INNOVATION & ERROR

**mutation:** a random change of an element in the genome. Here used to denote a random change in the value of a cultural trait.

The roots of the cultural evolutionary framework lie in genetic evolutionary theory, from which it inherits many concepts. A good example is **mutation**.

Most cultural transmission models will include a small chance (usually in the vicinity of 0.001) of a trait randomly changing its value. This may be coded as a separate process or included in the copying algorithm as a "copy error probability." In a cultural context, cultural mutation may represent an instance of individual learning (an innovation) where a new value of a trait is "discovered" as the emergence of a completely new cultural trait or a random change in the value of an existing one. Innovation can also mark a random loss of a trait. Depending on how cultural traits are represented, mutation may flip the value of a **Boolean** trait (e.g., 0 to 1) or modify it if the trait is represented as a continuous value (e.g., 1.10 changes to 1.09). In the latter case, the size of the change also needs to be specified, either as a fixed amount or by drawing from a distribution.

**Boolean:** a variable with only two possible values (e.g., TRUE or FALSE, 0 or 1, heads or tails). Also called a binary variable.

CODE BLOCK 5.22

```
to mutation-boolean
  ask turtles [
    if random-float 1 < 0.001 [
      ifelse cult1 = 0 [set cult1 1][set cult1 0]
    ]
  ]
end

to mutation-continuous
  ask turtles [
    if random-float 1 < 0.001 [
      set cult1 random-normal cult1 0.1
    ]
  ]
end
```

In the second example above, `random-normal` takes the current value as the mean and a standard deviation as a fixed parameter and returns a random number from within that probability distribution. In Xue (2013), the size of the cultural mutation (i.e., the standard deviation) was also modeled as a trait that was subject to mutation. This small shift resulted in some very unexpected dynamics and important conclusions regarding the evolution of cultural variability.

Parameters such as probability of mutation or the standard deviation can be set with a slider in the INTERFACE tab.

### CUMULATIVE CULTURAL EVOLUTION

So far all of the examples involved one or a few cultural traits passed from one individual to another. These basic building blocks can be used to construct highly complex models leading to cumulative cultural evolution. One of the most common ways of expanding the scope of these models is giving agents a string of traits or adaptations (think of this as a "cultural genotype").

CODE BLOCK 5.23

```
turtles-own [ cultgen ]
to setup
  ...
  crt n-agents [
    ...
    set cultgen n-values 5 [ i -> one-of [0 1] ]
  ]
...
]
```

Agents can then assess their fitness, similarity to each other, attractiveness, etc. on the basis of their cultural genotypes. To compare two lists where the position of each item is significant, we can use Hamming distance (Bookstein, Kulyukin, and Raita 2002).

**Hamming distance:** metric used to compare two sequences of the same length, which sums the number of differences in each position.

Imagine we have two lists representing two agents' cultural genotypes:

| 0 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|

| 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|

We can calculate the difference matrix by placing 1 whenever the two items of the list differ and 0 when they are the same.

Other similarity measures that can be used for comparing sequences are **Edit distance** and **Jaccard distance**.

| 1 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|

Hamming distance is the sum of these differences. In this example, it is equal to 2 (i.e., $1 + 1 + 0 + 0 + 0$) since only the first two elements of the lists differ from each other. In NetLogo this can be implemented as:

CODE BLOCK 5.24

```
to-report hamming-distance [ list1 list2 ]
  report length remove true (map [[?1 ?2] -> ?1 = ?2 ]
  list1 list2)
end
```

Hamming distance can be used to help agents determine whether they are close enough culturally to cooperate or whether the set of adaptations codified in the list is actually advantageous in a given environmental setting (here the cultural genotype would be compared to an ideal sequence representing a perfect adaptation). For example, Barceló et al. (2015) used a cultural genotype to model alliances and territoriality among Patagonian hunter–gatherers.

**Model:** *Patagonian Hunter–Gatherers* by Barceló et al.

This is only one of the many ways to model some aspects of cumulative cultural evolution—here we have provided key algorithms to serve as building blocks for more complex models. However, some recent reviews and applications of cultural transmission can help shed light on the approaches used here. Eerkens and Lipo (2007), for example, provide an argument for using cultural transmission models to understand changes in material culture over time. These approaches are employed by Crema, Kandler, and Shennan (2016) and simulated to examine the ways that design motifs of Neolithic pottery change over time. These simulations complement mathematical models by Kandler and Laland (2009) that examined changes in cultural diversity and look at the impact of independent invention on traits in a population. O'Dwyer and Kandler (2017) further examine neutral evolution's impacts on culture and simulate how critical innovation in the form of anti-novelty bias is in cultural evolution over time. Finally, Henrich (2001) suggests that biased transmission is responsible for most cultural change.

If it is the assemblages being compared, then the Brainerd–Robinson coefficient of similarity is a well-established choice (Peeples 2011; Peeples and Haas 2013).

You can find alternative implementations and other cultural transmission algorithms in the Model Zoo list at the end of the chapter.

**Model:** *Prestige Trade* by Graham

**Model:** *Cultural Trade* by Carrignon et al.

Turning back to the topic of trade, Graham (2005) has combined economic and cultural exchange frameworks into a model of economic exchange where agents build their reputation and only trade with partners of similar status (prestige bias). Carrignon, Brughmans, and Romanowska

(2020) similarly combined economic interactions with cultural transmission during which traders learn commercial strategies from each other. Cultural evolution studies is a rapidly expanding field in computational archaeology and other disciplines, such as ecology (Smolla et al. 2016). It is worth understanding the basic premises of the framework and following new developments. Much of the cultural evolutionary literature is based on numerical modeling rather than ABM, but with a little practice, most numerical models can be easily converted to ABM and expanded upon to address new questions.

## PUTTING TOGETHER A MODEL OF CULTURAL TRANSMISSION

To help you solidify many of the above concepts, we will examine a model by Premo (2014). His abstract model of cultural transmission examined the shifting patterns of artifact frequencies through time. The model follows a population where unbiased transmission occurs between generations of agents. The copying of cultural variants from the older generation to the younger generation occurs at random (i.e., it is not parent-to-offspring) at a rate of $1 - \mu$ with copying errors occurring with a probability of $\mu$. There is no horizontal transmission. At each step the older generation dies and a new generation is born, thus changing the frequency of different cultural traits, i.e., artifact types, in the population over time. Let's look at this model's code in a little more depth:

**Model:**
*Time-Averaging*
by Premo (2014)

**ABMA Code Repo:**
*ch5_teaching*

CODE BLOCK 5.25

```
to learn
  let teacher one-of turtles with [age = 1]
  set t1 [t1] of teacher
  ask teacher [set taughtThisGeneration
  (taughtThisGeneration + 1)]

  if random-float 1 < mu [
    set t1 nextNovelVariant
    set nextNovelVariant nextNovelVariant + 1
  ]
  set color t1
end
```

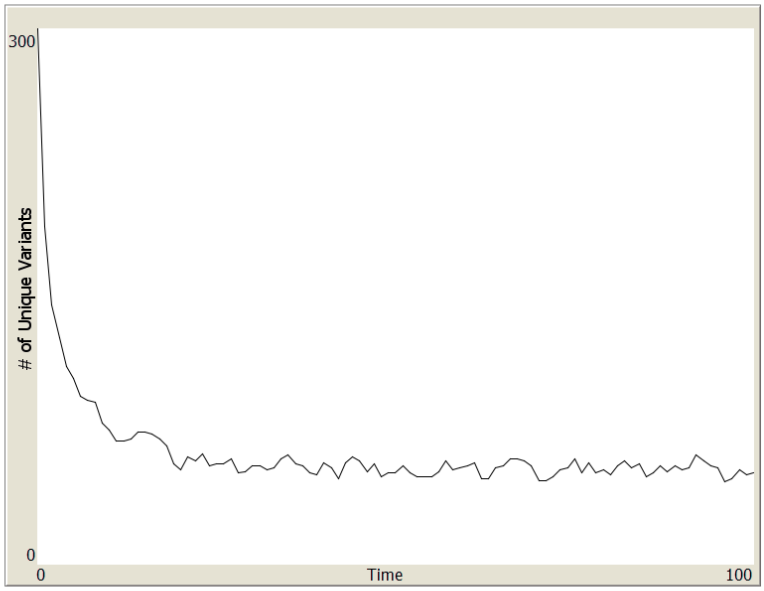



**Figure 5.7.** Premo (2014) shows that the number of unique variants, that is, cultural diversity, tends to decrease precipitously over time. While mutation allows for introductions of new traits, these do not persist, leading to an increase in homogeneity over time.

Each newly hatched agent picks a random `teacher` to copy a trait from. Then an agent has a probability of introducing a new variant of a novel trait due to a copy error with a probability $\mu$, which is set by a slider.

You can find several diversity measures, such as Shannon's index, in the original version of the *Time-Averaging* model.

If you set a low probability of mutation (i.e., $\mu$ less than 0.05) and any number of agents, we see that initially the cultural repertoire was highly diverse, as each had a unique variant. Under these parameter values, the model shows a dramatic decrease of diversity over time. Even with randomly mutated traits appearing in the population, they tend not to "stick." Instead, these traits tend to appear for one or a few time steps, as indicated in the "# of Unique Variants" axis of figure 5.7, and are rarely passed along for more than a few generations. These results may suggest that time averaging of traits can lead to similarly low diversity in assemblages from the process of unbiased transmission. Note that if you open this model in a version of NetLogo other than 5.0.2 (the version Premo used to write it), you will see a message about upgrading the model (fig. 5.8). Usually NetLogo takes care of the changes needed, but sometimes the differences between versions are too great. For older models, it's sometimes possible to upgrade to version 6.x.x by opening and saving them first in 4.1.x, then 5.1.x, then 5.3.x, and

**TIP**

We recommend installing multiple NetLogo versions as needed. One each of the last major versions 4.1.3 and 5.3.1 as well as a recent release of 6.x.x should cover it.

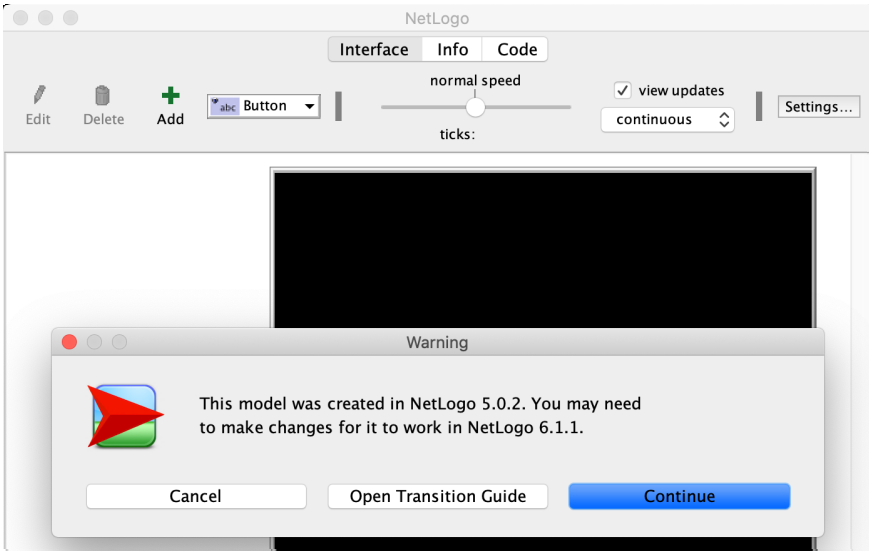



**Figure 5.8.** If you attempt to open an old model in a newer version of NetLogo, the software will give you this error message. Follow NetLogo's Transition Guide to ensure that the old model works in your newer version of the software.

only then in 6.x.x. If this doesn't work, see the NetLogo Transition Guide for help upgrading your models.[3]

## 5.3 The SIR Family of Models

Surprising as it may sound, one of the best ways to represent the spread of information over a population is with epidemiological models. Transmittable diseases and cultural traits share many similarities in the way they jump from one person to another. First, there needs to be a form of interaction for the transmission to occur—in epidemiology this usually requires a physical interaction, which until the onset of modern telecommunication technology was also the case for cultural transmission. Second, different variants have different transmission rates and therefore spread at different rates—the common cold and silly memes on social media share the quality of being easily transmissible. Finally, the rate of infection may depend on specific characteristics of groups of agents (e.g., age) which determines their susceptibility. For example, children tend to catch colds and memorize names of dinosaurs with more ease than their parents.

An unexpected feature of the COVID-19 pandemic is that we have all acquired an unprecedented level of understanding of epidemiological models. Many of their features (e.g., the $R_0$ number) can be adapted to studying cultural phenomena.

It's NetLogo convention to mark Boolean variable names with a question mark, such as `susceptible?`. This also applies to INTERFACE switches since they are either TRUE or FALSE.

[3] https://ccl.northwestern.edu/netlogo/docs/transition.html

The most commonly used family of models is based on a standard simulation framework known as the *SIR* (*Susceptible–Infectious–Recovered*) model (Weiss 2013). Sometimes you'll see these called *SEIR* models, with the *E* equating to "exposed" or some other variant. Originally expressed as a set of differential equations, the *SIR* framework can be easily translated into ABM, and still provides the base for many sophisticated epidemiological models today (e.g., Hammond et al. 2021).

**Model:** *Susceptible–Infected–Recovered (SIR)*

**ABMA Code Repo:** *ch5_SIR*

In an *SIR* simulation, agents exist in one of three states: Susceptible (healthy), Infectious (ill and spreading disease), or Recovered (dead, immune, or back into Susceptible depending on the characteristics of the particular illness) (fig. 5.9).

In a cultural context, we can translate this into "potential adopter of a trait," "carrier of a trait," and "rejector." To set up the model, you need a population of agents with three variables:

CODE BLOCK 5.26

```
turtles-own [ susceptible? infectious-timer recovered? ]
```

At each time step, a proportion of agents set with a `mobility` slider moves (`mobility`), and they can also transmit (infect) and recover. The key dynamics of this kind of model are the transmission and recovery mechanisms. This is the simplest representation:

CODE BLOCK 5.27

```
to move
    ask n-of floor (mobility * count turtles) turtles [
    rt random 360
    fd 1
  ]
end

to infect
  let carriers turtles with [ infectious-timer != 0 ]
  ask carriers [
    let susceptible-neighbors turtles with
    [ susceptible? ] in-radius 2
```

You can use even this basic model to see how agent mobility has a dramatic impact on the shape of the infection curve and the final outcomes.

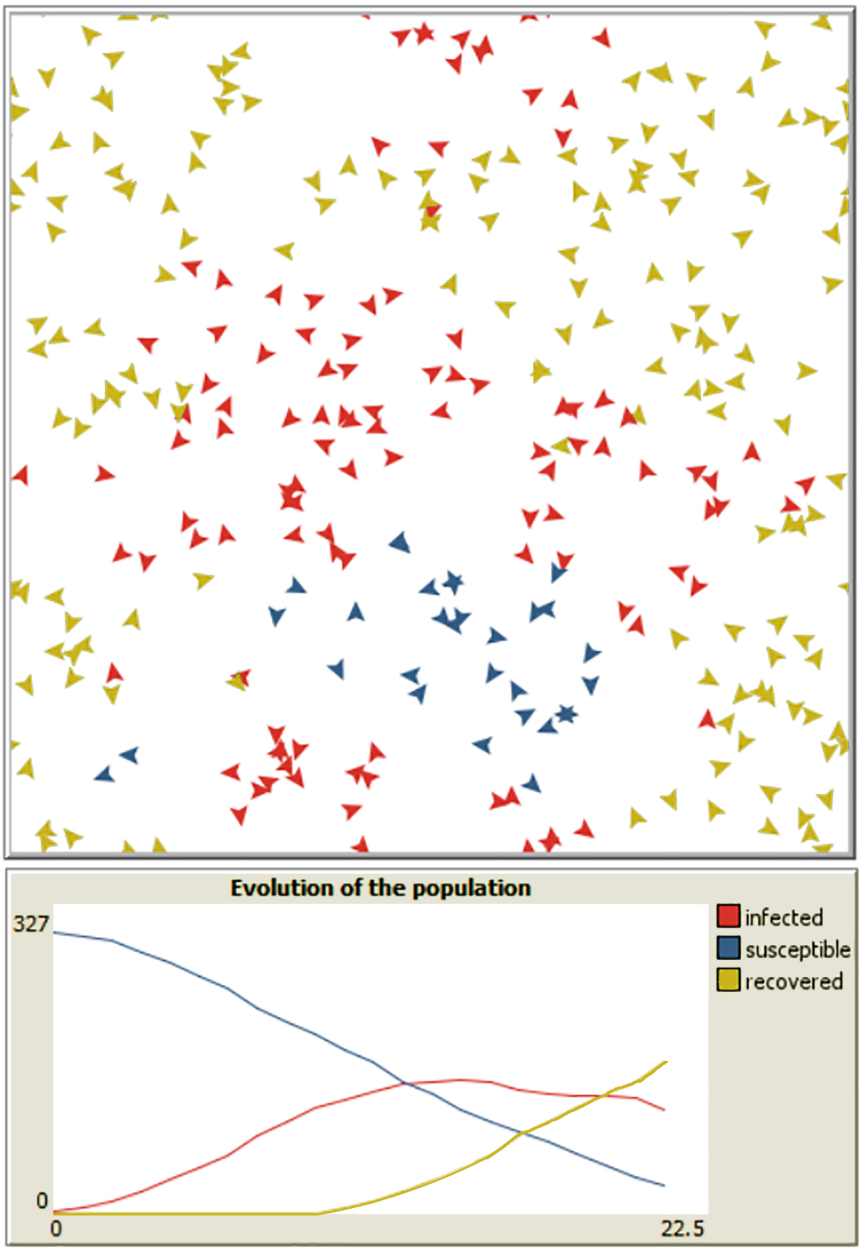



**Figure 5.9.** The evolution of an epidemic. Initially the population of infected agents increases rapidly. The system then crosses through an inflection point and the transmission decreases when there is insufficient susceptible population to infect. In this case the decrease is due to acquired immunity but other interventions, e.g., vaccines, can similarly reduce the proportion of susceptible individuals.

CODE BLOCK 5.27 (cont.)

```
    if any? susceptible-neighbors [
      ask one-of susceptible-neighbors [
        set susceptible? False
        set infectious-timer 10
        set color red
      ]
    ]
  ]
end

to recover
  ask turtles with [infectious-timer > 0][
    ifelse infectious-timer > 1 [
      set infectious-timer infectious-timer - 1
    ][
      set infectious-timer 0
      set recovered? True
      set color yellow
    ]
  ]
end
```

To modify the way transmission and recovery proceed, the modeler could add a probability of transmission and recovery to be, for example, randomly distributed throughout the population, or dependent on another variable (such as age, popularity, utility, etc.). The cultural transmission algorithms discussed above are of great use to represent the particularities of the studied process. Similarly, distributing agents on a network instead of along continuous space can change the rate and other characteristics of the spread. You will find these extensions implemented in several models in the NetLogo Models Library, such as, *epiDEM Basic*, *Spread of Disease*, or *Virus on a Network*.

Most *SIR* models include a probability, for infected agents, of dying. Try adding an age-specific susceptibility that limits who is open to learning new things. How does this affect the transmissibility of new innovations?

The *SIR* family of models is a highly flexible framework that can be adjusted to almost any problem involving transmission of information at a large scale. Importantly, the concepts involved are quite intuitive,

and with the recent increase in reporting on epidemiological models in the media, the general understanding is high even among nonexperts. This makes it relatively easy to explain its dynamics to a general social science audience. A pioneering example of the application of a simple *SIR* model to archaeology is Graham (2006). The author released information on a transport network constructed on the basis of the Antonine Itineraries to investigate the rate of information spread through different Roman provinces. The simulation revealed how in some provinces (e.g., Iberia, Gaul) information spread at a different rate than in others due to the structure of their transport networks.

**Model:**
*The Antonine Itineraries*
by Graham

## 5.4 Ontology Building & Parsimony

A critical aspect of model development, as important as coding and debugging, is learning how to choose what to code into the model and what to leave out. It is a challenging process, and ideally you want to have the full **ontology** described before you even start to code. In practice, it often develops throughout the project with elements being added or removed until the last moment before running the simulation.

**ontology:** the full representation of the model's world, including names and definitions of entities and their categories and properties, the relationships between entities, and the rules of behavior.

There are two aspects that can help us decide what the artificial world we create should include. The first is your research questions, which should be well developed before you sit down to code. The second is the principle of **parsimony**. We will look at them in turn.

**parsimony:** the idea that a model should have as few details as possible while still representing the modeled phenomenon. Often defined as the Occam's razor principle.

A model is by definition a simplification of reality. It follows from this that any one system can be represented in a multitude of different models, none of which is more "correct" than the others. Think of a city and how it could be depicted on a map as a series of streets and buildings, or as a density map of different economic activities, or as a network of people who live in it. Ultimately, how we want to represent the system will depend on what we want to learn from that representation. If we are interested in traffic and congestion, then the streets/buildings map probably best meets our goals. Less so if we use an *SIR* model to evaluate the most effective epidemic controls for an urban area. Thus, when deciding what to include in your model, the research questions are the determining factor.

Research questions are sometimes compared to a knife used to carve the simulation into the desired shape and size.

The principle of parsimony means that we should strive not to overcomplicate the model to produce the desired output. For example, we could

imagine a trade model where including a number of goods, each with a specific quality and price, might increase the realism of the model. Likewise, we could add the transport cost, divided between the land and the sea routes as well as the season. But would these changes, in fact, enable us to understand the dynamics of commercial exchange better? Or would they just add meaningless noise? It is also often difficult to establish where to stop in the quest for more realism. What about transport on donkeys versus carts? Or tableware traveling as part of foodstuff cargo? In the Brughmans and Poblome (2016) model, the transport cost was not accounted for explicitly, yet the authors demonstrate convincingly that the tableware data collected over the eastern part of the Roman Empire indicates a relatively high level of economic integration in the region.

Even a simple model with 5 parameters where 3 values need to be tested for each would require 243 runs. Times 100 to account for stochasticity in the model, and we're looking at weeks of experiment runs clogging your computer.

We could go on forever adding more and more details to the simulation. The answer to "Could you make the model do $X$?" is always yes, but the price of more details comes at the stage of experiment design when we need to run all combinations of parameter values. The number of scenarios tends to increase exponentially, making it difficult not just to properly sweep through the parameter space but also to understand and interpret the results (see ch. 9 for experiment design).

Instead, by simplifying the model down to the most parsimonious elements, we can demonstrate the most significant mechanisms that guided the whole system. In future iterations of the model, we can then add new factors, behaviors, or specific data and explore the dynamics further. However, we will only be able to assess their impact if we have a baseline, that is, the most parsimonious model with which to compare these more evolved versions.

Another way of thinking about which of all possible elements to include within a model is to decide what set of model elements is both **necessary** and **sufficient** to explain the phenomenon you want to study. Each element should be necessary in the sense that if you removed it, the model results would no longer answer your research question. Moreover, the combination of elements in the model needs to be sufficient to produce results that are comparable to the archaeological record. For example, adding subsistence strategy to a cultural transmission model of changes in pottery decorations is probably not necessary, but removing family ties could result in

a model that is not sufficient to differentiate between vertical vs. horizontal transmission.

Upholding principles of parsimony—keeping a model as simple as possible—is one of the best practices of agent-based modeling. By stripping away unnecessary complexity, your model will be easier to examine and easier to reproduce, and can be built upon when new questions arise. In this way, you can see how building simple models of movement, exchange, or subsistence—as we did in the first section of the book (chs. 1–3)—can provide a parsimonious backdrop for many questions related to archaeology: How do exchange mechanisms influence trade, mobility, or demography? The algorithms in this part of the book (chs. 4–6) can be used to build your own models from such a simple base.

This is one of the best practices of agent-based modeling (known as KISS; see Axelrod 1997).

## 5.5 Summary

In this chapter we looked at the many different ways in which economic and cultural exchange can be modeled. The fields of economy and cultural evolution have produced many extensions to the basic algorithms we have presented here. A deep dive into the literature is warranted whenever a model involves transmission between agents, be it of economic nature or of information. Economy, in particular, is a discipline with long-standing theories, and it is likely that a particular type of commercial interaction you would like to model already has a large body of theory and literature behind it. Finally, we touched on one of the most challenging phases of model development: ontology building. It is one of the most critical pieces of model development. Devising a clear ontology early on will enable better code development and experiment design and analysis, and it will make your task of communicating your results to your audience much easier.

In the next chapter, we will return to the topic of subsistence and explore different foraging and harvesting algorithms. We'll also talk about how to parameterize your model and where to look for realistic parameter values. As you continue, keep in mind the practice of finding theoretical and formal models from other disciplines, as well as the need for well-defined ontology as we begin to parameterize our models. A clear road map will make picking parameter values much easier.

## End-of-Chapter Exercises

1. In chapter 3 agents collect resources. Extend the model by allowing them to trade with each other.

2. Now, let them record locations in which they have come across the highest density of resources. They can now exchange this information with each other. Use different cultural transmission algorithms to model that process.

3. If you're up for a challenge, try using the previous examples and algorithms developed so far to implement a classic model by Geertz (1978). In this model, individuals will establish a relationship with a seller. How does long-term relationship tracking change the economic model?

4. In chapter 4 we suggested an end-of-chapter exercise combining multiple models. Take that exercise and strip it down to the most parsimonious model to answer a question at hand. Is it difficult to remove pieces you already added to your model?

## Trade & Exchange Model Zoo

▷ **The Antonine Itineraries**

S. Graham. 2006. "Networks, Agent-Based Models and the Antonine Itineraries: Implications for Roman Archaeology." *Journal of Mediterranean Archaeology* 19 (1): 45. doi:10.1558/jmea.2006.19.1.45

Code:
http://www.graeworks.net/2013/05/16/antonine-itineraries-abm/

▷ **Cultural Trade**

S. Carrignon, T. Brughmans, and I. Romanowska. 2020. "Tableware Trade in the Roman East: Exploring Cultural and Economic Transmission with Agent-Based Modelling and Approximate Bayesian Computation." *PLOS ONE* 15, no. 11 (November): e0240414. doi:10.1371/journal.pone.0240414

Code: https://osf.io/s5mdw/

▷ **Market, Edgeworth Box Game, and Shops**
L. Hamill and N. Gilbert. 2016. *Agent-Based Modelling in Economics.* Chichester, UK: Wiley. doi:10.1002/9781118945520

Code: https://hamill.co.uk/lynne-hamill/abm-in-economics/models

▷ **Mercury Roman Trade**
T. Brughmans and J. Poblome. 2016. "Roman Bazaar or Market Economy? Explaining Tableware Distributions in the Roman East through Computational Modelling." *Antiquity* 90 (350): 393–408. doi:10.15184/aqy.2016.35

Code: https://www.comses.net/codebases/4347/releases/1.1.0/

▷ **Prestige Trade**
S. Graham. 2005. "Agent-Based Modelling, Archaeology, and Social Organisation: The Robustness of Rome." *The Archaeological Computing Newsletter* 63:1–6

▷ **Simple Economy**
U. Wilensky and W. Rand. 2015. *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo.* Cambridge, MA: MIT Press, April.

Code: https://ccl.northwestern.edu/netlogo/models/SimpleEconomy

▷ **Susceptible—Infectious—Resistant (SIR)**
C. Yang and U. Wilensky. 2011. *NetLogo EpiDEM Basic Model.* Evanston, IL.

Code: https://ccl.northwestern.edu/netlogo/models/epiDEMBasic

▷ **Time-Averaging**
L. S. Premo. 2014. "Cultural Transmission and Diversity in Time-Averaged Assemblages." *Current Anthropology* 55, no. 1 (February): 105–114. doi:10.1086/674873

Code: https://www.journals.uchicago.edu/doi/full/10.1086/674873