The Importance of Ontological Structure: Why Validation by 'Fit-to-Data' Is Insufficient

Chapter *in* Understanding Complex Systems · November 2017

DOI: 10.1007/978-3-319-66948-9_8

CITATIONS
14

READS
497

2 authors:

J. Gary Polhill
James Hutton Institute
155 PUBLICATIONS **5,358** CITATIONS
SEE PROFILE

Douglas Salt
James Hutton Institute
11 PUBLICATIONS **37** CITATIONS
SEE PROFILE

Preprint of chapter 8 (pp, 141-172) in Edmonds, B. and Meyer, R. (2017) (eds.) Simulating Social Complexity, 2nd Edition. Springer https://doi.org/10.1007/978-3-319-66948-9_8

The importance of ontological structure: why validation by 'fit-to-data' is insufficient

Draft chapter for Simulating Social Complexity
Gary Polhill and Doug Salt
19 May 2017

Why read this chapter? When you have built an agent-based model, you need some way of assessing how 'good' it is. We will tell you how this is done traditionally in empirical contexts, through measures of fit-to-data, but also argue (using neural networks as a baseline) that fit-to-data is not enough in the kind of situation where agent-based models are useful: you also need to assess the model's ontological structure. Hence, you need to know what the ontological structure is, exactly, how to assess it, and whether and if so how it can be traded off against fit-to-data. These matters are much less settled than is the case for measures of fit-to-data, but we will give you an overview of the state-of-the-art.

Abstract:

This chapter will briefly describe some common methods by which people make quantitative estimates of how well they expect empirical models to make predictions. However, the chapter's main argument is that fit-to-data, the traditional yardstick for establishing confidence in models, is not quite the solid ground on which to build such belief some people think it is, especially for the kind of system agent-based modelling is usually applied to. Further, the chapter will show that the amount of data required to establish confidence in an arbitrary model by fit-to-data is (big data aside) often infeasible. This arbitrariness can be reduced by constraining the choice of model, and in agent-based models, these constraints are introduced by their descriptiveness rather than by removing variables from consideration or making assumptions for the sake of simplicity. By comparing with neural networks, we show that agent-based models have a richer ontological structure. For agent-based models in particular, this richness means that the ontological structure has a greater significance, and yet is all-too-commonly taken for granted or assumed to be 'common sense'. The chapter therefore also discusses some approaches to validating ontologies, though the state-of-the-art is far from a situation in which there are established standards.

Introduction

Neural networks are universal function approximators (Hornik et al. 1989). This means that given a set of data, they can approximate it to within an arbitrary degree of accuracy simply by adding more parameters. Though it may seem strange to compare neural networks with agent-based models for the purposes of validation and generalization, there are useful lessons from so doing that illustrate where agent-based models add value to traditional modelling approaches, and why validation is not so straightforward. The main contrast

between neural networks and agent-based models comes down to the 'ontology'. Essentially, apart from the labels assigned to the input and output units of a neural network, neural networks don't have an ontology at all. What they do have is a mathematical structure that allows the number of parameters to be arbitrarily varied, and with that, arbitrary degrees of fit to a set of data to be achieved. By contrast, agent-based models have a rich and highly descriptive ontology, but, like neural networks, potentially have a large number of parameters that can be varied (especially if we consider each agent uniquely). In this chapter, we examine some approaches to validation and generalization in neural networks and consider what they tell us about agent-based modelling. Our arguments are that validation needs to look beyond the relatively trivial question of fit-to-data, especially in non-ergodic complex systems. Rather than being a weakness of agent-based modelling, the challenges of validation and generalization point to its strengths, especially in social systems, where the language used to describe them is influenced by evolving cultural considerations.

Introduction to neural networks

In this section, we will briefly explain what neural networks are, the mathematical formulas that underpin them, and the way they are structured. The main points we wish to introduce are that, though neural networks have tremendous potential to approximate data, there is nothing about their structure or the mathematics underpinning their functioning that necessarily reflects any structure or mathematics in whatever system the data were taken from.

Neural networks were originally conceived as simulations of brains, but are essentially networks of nonlinear functions with parameters that are adjusted according to a learning rule. There are several different kinds of neural network mathematically speaking, and for each kind, there can be several different learning rules and minor adaptations and variations thereof. Biologically, a neuron is a cell with axons connecting it to other neurons. In an agent-based simulation of a brain, we would simulate a neuron as an agent, and an axon as a link. The behaviour of the neuron is simply to emit an electrical pulse periodically. The more frequent the pulse, the more 'excited' the neuron. Connections between neurons can be excitatory, or inhibitory. An excitatory connection means that there is a positive relationship between the excitation of the two connected neurons, all other things being equal – one neuron's excitement increases that of the other. An inhibitory connection means that the relationship is negative – one neuron's excitement decreases that of the other. The connection has a strength – the stronger the connection between one neuron and another, the more significant the relationship is in comparison with other neurons the neuron is connected to.

When simulating neurons, the pulsation is ignored, and the frequency of pulsation modelled as a variable. Simulated neurons are typically called *nodes*. The axons form the links in a directed graph connecting the nodes, and the directedness means that nodes have input axons and output axons. Simulated axons are typically called *weights*, largely because it is the value of the weight (representing the strength of the connection) that is of primary interest. The

weights of a neural network are its parameters, and the job of the learning algorithm is to determine their values.

The qualitative description of the behaviour of neurons is of course given a precise mathematical specification in simulated neural networks. Though there are variants, typically the *excitation*, x_j, of a node j is given by the weighted sum of its inputs [1]:

$$x_j = \sum_{i \in \text{inputs}} w_{ij} o_i \qquad [1]$$

where o_i (usually in the range [0, 1], though some formalisms use [-1, 1]) is the output of a node i with a connection that inputs to node j, and w_{ij} is the strength (weight) of that connection.

Nonlinearity of the behaviour of the node is critical to the power that the neural network has as an information processing system. It is introduced by making the output o_j of a neuron a nonlinear function of its excitation x_j. There are a number of ways this can be achieved. Since many learning algorithms rely on the differentiability of the output with respect to the weights, the sigmoid function is typically used:

$$o_j = \frac{1}{1 + \exp\left(-x_j\right)} \qquad [2]$$

So, a neural network essentially consists of a directed graph of nodes, where each of the links has a weight. If the graph is acyclic, the neural network is known as a *feed-forward* network. (If cyclic, the network is *recurrent*.) Nodes with no input connections are *input* nodes; those with no output connections are *output* nodes. Since they have no input connections and hence no excitation, input nodes are often also not given a nonlinear treatment as per [2], though this breaks somewhat with the simulation of a neuron. Similarly, nonlinearity may not be applied to output nodes. If there are N input nodes, and M output nodes, then essentially a feed-forward network without nonlinearity on the output nodes is computing a mapping from $\mathbf{R}^N$ to $\mathbf{R}^M$. With nonlinearity, the mapping is from $\mathbf{R}^N$ to $[0, 1]^M$.

A further simplification of the structure of the network is to arrange the nodes into distinct layers. (It can be proved that this does not lead to loss of potential functionality.) In a layered feed-forward neural network, no pair of nodes in the same layer is connected to each other, and all nodes in each layer have the same sets of input and output nodes: the input nodes being all nodes in the layer below, and the output nodes being all nodes in the layer above. There is one exception. Every node has an input connection from a *bias* node. (If there is no nonlinearity on the input nodes, then they have no input connection from the bias node.) The bias node always has an output of 1, and the weights on its fan-out connections enable the nodes to require different levels of excitation before they transition from having output close to 0 to having output close to 1.

This simplification means that the choice of network structure is simply a question of determining the number of layers, and for the layers that are not input or output layers (the so-called *hidden* layers), the number of nodes to use in each layer. The number of nodes in the input and output layers is of course determined by the dimensionalities of the domain and range of the function to be approximated. It has been proved (Cybenko 1989; Funahashi 1989; Hornik et al. 1989) that one hidden layer is sufficient to approximate any function. Although having more hidden layers can mean that the contribution of the weights closer to the input units to the difference between the actual and desired output of the network is more diluted, it can also be shown that more efficient network topologies (in terms of number of weights) involving two hidden layers can achieve the same level of accuracy that can be achieved with one hidden layer (Cheng and Titterington 1994; Chester 1990).

The mathematically literate reader with an interest in geometry will have realized that each node represents a hyperplane in the space of the layer below, on one side of which the node tends towards an output of 1 the greater the perpendicular distance from the hyperplane, and on the other side the node tends towards an output of 0. Nodes in the first layer therefore perform various bisections of the input space. Nodes in the second layer perform various bisections of the unit hypercube formed by the output space of the nodes in the first hidden layer, and so on. It can be shown that two hidden layers are sufficient to realize an arbitrary separation of input space into regions where output units have outputs close to 1 or 0 (Lippmann 1987).

The algorithms used to determine the weights such that the network as a whole provides a good fit to data are not particularly of interest here. This material is covered in various introductory textbooks on neural networks (e.g. Bishop 1995; Gurney 1997; Hertz et al. 1991). What is of interest is that, having seen the structure of a neural network and what it does, it is immediately clear that there is nothing in that structure that reflects the real world, except for the assignment of input nodes and output nodes to specific variables in the data to be fitted, and, to a debatable extent, in the number of hidden nodes and layers, which essentially reflect how complex the modeller expects the function to fit the data to need to be. Neural networks have the absolute minimum in the way of ontological structure it is possible to have.

Calibration, validation and generalization in neural networks

Calibration, validation and generalization are three steps in the development and application of any model. We discuss them here in relation to neural networks, first with a view to clarifying what we mean by those terms, second to discuss some of the ways in which generalization (the application of the model) can go wrong even for a well-validated model.

Since various terms are used in the modelling literature for the three processes intended here by the words ‘calibration’, ‘validation’ and ‘generalization’, it is best to be clear what is meant. The process begins with a set of data, with some explanatory (input) variables and response (output) variables, and a model with a predefined structure that has some parameters that can be adjusted. The data

are split into two not necessarily equal parts. The larger part is typically used for *calibration*: adjusting the parameters so that the difference between the model's prediction for the input variables in the data and the corresponding output variables in the data (the *error*) is minimized. In neural networks, this is referred to as *training*, and entails adjusting the values of all the weights.

There is a caveat to the use of the term 'minimization'. For reasons such as measurement error in the data, if a function is capable of providing an exact fit to the data, this is potentially undesirable, and is seen as '*overfitting*'. So, when we say we want to minimize the error, it is usually understood that we wish to do so without overfitting.

Bearing this in mind, at the end of the calibration process, you have a parameterized neural network with all the weights specified that you now want to be able to use to make predictions with. Except, of course, you want to have some degree of confidence in those predictions. *Validation* is the process of developing that confidence, and it is achieved by using the data you kept aside and didn't use during calibration to estimate how good your future predictions will be. So, having reached a point where you are happy with the error on the calibration data, you use the validation data to tell you how confident you should be in the model you have fitted: the error rate on the validation set is an estimate of the expected error rate for prediction.

Generalization is the ability of the model to provide output for untrained input. There are two aspects to this. The first is whether the input can be represented using the formalism provided by the model. In the case of neural networks, the question seems simply to be whether the input can be adequately expressed using the same set of dimensions and any encoding thereof as the data used for calibration and validation. It may seem unfair to expect a model to be able to provide output for cases that cannot be expressed using the 'language' the model was built with. However, sometimes, arguably, that is what happens. Measures of inflation, for example, are based on a 'basket of goods' that changes from time-to-time as people's buying habits change. This change arguably changes the meaning of inflation. Though something of a straw man, if you have calibrated a model using a measure of inflation that uses one basket of goods, and then naively expect it to give meaningful output for a measure of inflation that uses another, then perhaps you are expecting the model to provide output for cases that cannot be expressed using the language the model was built with.[1] Similar problems exist with other social statistics that might be used as variables input to or output from a model, particularly where there are changes in the way the variables are measured from one region to another.

A second problem comes from what you left out of the model when you first built it. Although this too may seem like an unfair criticism, perhaps when you built the original model, a particular variable was not included as an input variable

[1] Less naively, you would use a calculated inflation figure for the old basket of goods as input to the model; however, if people are not buying things in the old basket, the model may still not be providing meaningful output.

because it was not seen as having any significant relationship with the output. Since the model was calibrated and validated, however, a large change in the ignored variable might have occurred that has affected the relationships between the variables you did include. So, although when you come to compute a prediction for a new input you have all the data you need, and can perform the computation, really, the values for the variables you have as inputs to your model do not adequately reflect the scenario any more. This is known as 'omitted variable bias' in the econometrics literature (see, e.g. Clarke 2005).

A final problem is a consequence of encoding variables that have nominal values. Assuming an appropriate encoding of nominals in the input variables of the model, the calibration and validation data may only have provided a subset of the nominals the variable can have. The generalization may, however, be for a value of the nominal that was not in the data used to construct the model. For neural networks, this is less of an issue than with symbolic AI machine learning algorithms: one of the supposed advantages of neural networks is that they are less 'brittle' with respect to the language of representation of states of the world, because they do not rely on the language having a specific vocabulary to represent every possible state that might ever be of interest (Aha 1992; Hanson and Burr 1990; Holland 1986).

In essence, calibration is the process of finding the parameters of a neural network (or more generally, any model) that best fit your data. Validation is the process of establishing the confidence you can expect to have in the predictions of the model based on the data you have got. Generalization is the capability of a model to make predictions in new situations. There are various reasons why that capability may be questioned. Apart from the relevance of the data used for calibration and validation in the new context, the reasons relate to how the modeller chose to encode, or represent, the data.

Bias versus variance

The representation of the data is not the only choice the modeller makes. This section covers the dilemma a modeller faces when choosing the structure of the model. In the case of neural networks, that structure is the number of layers and hidden units, which collectively determine the number of weights, or parameters the model has. The fewer the number of parameters, the easier the model is to calibrate, but there is a risk of oversimplification. Since it is so easy to add more parameters to a neural network, there is a temptation to add more parameters. We introduce some rather advanced mathematics (Vapnik-Chervonenkis theory) to argue that in terms of demand for data, adding more parameters can be exponentially costly.

Not all approaches using mathematical functions are ontology-free in the way neural networks are. If we are modelling oscillatory systems, for example, we might start with trigonometric functions. In general, the set of functions we are willing to consider for modelling a system constitutes our 'bias' – the smaller the set of functions, the greater the bias. Even neural networks have a 'bias' (*not* to be confused with the 'bias' node in the network itself), which is inversely related to the number of parameters (weights) in the network. In the ideal world, we

would have a very high bias that constrained the set of functions we would consider so much that calibration, the search for 'the' function we are going to accept as modelling the target system is trivial. The price to pay for this bias is that the data may not fit very well to the set of functions we are willing to consider; if we were only willing to expand that set of functions more, we would be able to achieve a much better fit to the data. The opposite of this meaning of 'bias' is 'variance'; in neural networks, this variance is directly related to the number of weights in the network. High-variance models can be adjusted using the parameters to realize a wide range of input-output mappings, with the obvious cost of increasing the volume of search space in which to find the optimum such mapping.

Introducing bias just to make the modelling process feasible is arguably unscientific: you are allowing your chosen modelling technique to drive your analysis of a system, rather than allowing your knowledge of that system to determine the way you describe it in your model. This kind of unscientific bias is one of the practices that has led some in the agent-based modelling community to be critical of making assumptions 'for the sake of simplicity' (e.g. Moss 2002; Edmonds and Moss 2005). Although some of these criticisms are focused on the infeasibility of the analysis itself were a more realistic representation used that did not make simplifying assumptions (e.g. the computation is undecidable), the feasibility of an empirical modelling process does depend on the availability of data.

Like neural networks, agent-based models potentially have large numbers of parameters – a multiple of the number of agents and the number of links in the social network. These parameters determine the heterogeneity and interaction dynamics of the model. For more traditional modelling paradigms, having large numbers of parameters is regarded with suspicion. From a practical perspective, there is a good reason for this heuristic: a high-variance model is more challenging to calibrate. Each dimension of parameter space adds exponentially to the scale of the search task, and to the requirement for data. Another reason is an interpretation Ockham's Razor in a modelling context: if I have two models with the same behaviour, I prefer the one with fewer parameters. Ockham's Razor is often stated as *entia non sunt multiplicanda praeter necessitatum* (literally: entities should not be multiplied more than necessary; or more naturally, explanations should not use unnecessary entities) – were it not for the qualifier, this statement would be the antithesis of agent-based modelling! (The case for agent-based modelling being that it *is* necessary to represent all the agents if you want to understand the emergent system-level dynamics.)

However, the orthogonality of the parameters in agent-based models may be more questionable than in traditional mathematical models. Essentially, in traditional mathematical modelling, each parameter is contributing to the potential 'wiggliness' (to use a term from the spline literature, e.g. Wood and Augustin 2002) of the function the model realizes. Though it is possible (e.g. Gotts and Polhill 2010), it is not necessarily the case that having another agent in the system will mean that the dynamics of the system as a whole are hugely different; adding another connection in a neural network, by contrast, does

increase the 'power' of its function to realize different shapes in the mapping from input to output by adjusting the weights. The suspicion of traditional mathematical modellers towards agent-based models because of the apparently large number of parameters may therefore not be justified, but further research is needed to establish this.

There may be a way to assess the question of the 'power' a system of interacting agents has to realize different 'shapes' from input to output (however that is understood in an ABM context) quantitatively. In the early 1970s Vapnik and Chervonenkis (1971) published a paper that provided a lower bound on the probability that the difference between the actual predictive power of a classifier system and that estimated from calibration is more than a small amount, based on the amount of data it is given, and something called the 'Vapnik Chervonenkis dimension' of the classifier. The inequality is written thus (ibid., p. 269):

$$P(|g - h| > \varepsilon) \leq 4m(2n)e^{-\varepsilon^2 n/8} \qquad [3]$$

Where g and h are the actual and estimated generalisation ability respectively (the proportion of instances that are correctly classified), ε is the small amount we want to bound the difference between g and h to, n is the amount of data (as number of instances), and $m(x)$ is a function that tells you the number of different realizations the classifier can make on x datapoints. The function $m()$ is equal to 2^x until $x = d_{VC}$, the Vapnik Chervonenkis (VC) dimension of the classifier, after which it continues to grow, but at a polynomial rate less than 2^x and no more than $x^{d_{VC}} + 1$ (Hertz et al. 1991, p. 154). A rough idea of the shape of the growth function $m()$ can be seen in Figure 1, particularly the red (top) curve when $d_{VC} = 4$. In a log-log plot, $m()$ is convex until a critical point at which it becomes linear; as stated above, this critical point is the VC dimension of the function d_{VC}, but the red curve in Figure 1 is $4m(2n)$, so in fact the critical point on the red curve should be at $n = 0.5d_{VC}$. However, since $x^{d_{VC}} + 1 > 2^x$ for lower values of x, the polynomial upper bound on $m()$ isn't informative; the critical point in Figure 1 at which $m()$ becomes linear is therefore higher than would otherwise be expected.

To understand [3] a bit better, imagine $\varepsilon = 0.01$. That means you want the difference between the actual and estimated abilities to be less than 0.01 ideally. So, suppose you have a validation ability (h) of 0.95 (5% of the model's predictions on the validation data are wrong), then with $\varepsilon = 0.01$, you are saying you want your future predictions to have an ability (g) in the range [0.94, 0.96]. How certain do you want to be that you have achieved that? Suppose you want to be at least 99.9% certain; so one in a thousand predictions will have an ability outside the above range. Then you want the probability on the left hand side of [3], P, to be 0.001. How can you achieve this? The right hand side says that the probability can be reduced by using a function with a smaller VC dimension (so $m(2n)$ is smaller), using more data (increasing n), or being less fussy about you close your validation ability is to the ability you expect in future predictions (increasing ε). To achieve a probability bound of 0.001, you need $\exp(\varepsilon^2 n / 8)$ to be at least a thousand times more than $4m(2n)$.

Mapping an ABM context to a classifier one would be somewhat awkward, though we could ask under what conditions (these conditions being the 'input space') the ABM produces a certain outcome – an outcome that either happens or doesn't. However, there is the additional problem that any stochasticity in the model will possibly generate different outcomes given the same conditions. Provided these issues can be addressed, given a thorough exploration of the ABM's parameter space, we may be able to estimate the VC dimension of the model given such an interpretation of its behaviour. We could then see the difference that adding another agent had, and compare both with adding a parameter to a neural network, where approaches to estimating the VC dimension or computing it directly have already been investigated (e.g. Abu-Mostafa 1989; Watkin et al. 1993).

One of the rather depressing consequences of using the VC formula is that the value of n needed to get P down to an acceptable level turns out to be rather high, even for models with quite low VC dimension. Figure 1 plots expressions in [3] on a log-log scale, using the x-axis for n, the amount of data. The coloured curves show upper bounds for $4m(2n)$, and the black curves show P / $\exp(-\varepsilon^2 n$ / 8) for ε in each of {0.05, 0.01, 0.001} and P = 0.001. The intersections of the black and coloured curves show the values of n (on the x-axis) at which P in [3] has an upper bound of 0.001. For example, if d_{VC} = 2 (cyan curve), and ε = 0.05, then n needs to be roughly 10^5 for P to have an upper bound of 0.001. For quantitative social data, that would be a very simple model for a very expensive questionnaire.

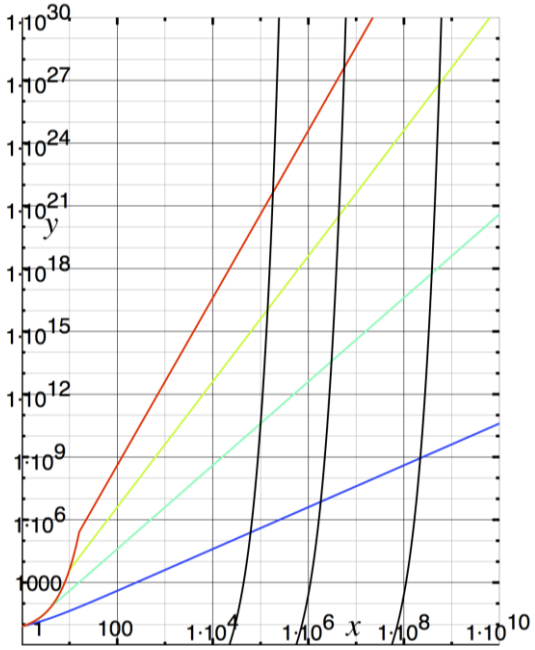

Figure 1. Plots showing the two expressions in [3]. Coloured curves are upper bounds 4m(2x) for d_{VC} in {1 (blue), 2, 3, 4 (red)}, The black curves show 0.001/exp(-ε²x / 8) for ε in {0.05, 0.01, 0.001} (left to right, respectively).

These high estimates are partly a consequence of the fact that the VC formula and growth function $m()$ are both upper bounds. However, the high estimates are also a consequence of the function under scrutiny essentially being an

arbitrary choice, without any other information about the system the data have come from or the way the model describes that system. The VC formula is therefore very much a 'worst case', but one that applies to neural networks insofar as relatively little information about the system is encoded in the network's topology. That information is essentially the modeller's assumptions about the appropriate level of 'wiggliness' needed to fit the data – which may be as much about the pragmatics of training the network and the amount of data available as it is a reflection of the system the data have come from.

Using knowledge to constrain the choice of model is one way to reduce the VC estimate. Traditionally, this might be achieved effectively by reducing the VC dimension of the set of models being considered, using the kind of practice criticized above for being 'unscientific'. Introducing bias by removing variables from consideration, reducing the number of parameters on terms using those variables (e.g. by only considering linear models), or making other over-simplifying assumptions is, however, not the only way that we can constrain our choice of model. Though the impact on the VC dimension is less clear, in agent-based models, we can also constrain our choice of model by making it more 'descriptive' (Edmonds and Moss 2005). This essentially amounts to appropriately tuning the model's 'ontology' or 'microworld', but before considering the ontology in more detail, since agent-based models are typically applied to complex systems, we will consider some arguments about validation by fit-to-data in such systems.

Complex systems and validation by fit-to-data

Since agent-based models are applied to complex systems, this section introduces an important article (Oreskes et al. 1994) posing arguments about the degree to which we should trust fit-to-data as a measure of our confidence in a model's predictions in complex open systems. We move on to criticize Ockham's razor – a heuristic often used by modellers to give preference to simpler models with the same fit-to-data, and one that has already been argued against on different grounds by Edmonds (2002).

Naomi Oreskes and colleagues (1994) have argued eloquently that environmental systems (and hence socio-environmental systems) are 'open', and hence traditional validation expressed as fit-to-data commits a logical fallacy when used as a basis to judge the degree-of-belief we should have that a model is a 'good' one. Essentially, the fallacious argument affirms the consequent by starting with the observations that

- good models fit the data ($G \subset F$), and
- my model fits the data (F);

and concluding that

- my model is a good model ($\vdash G$).

Oreskes et al. (1994) assert that (prejudices such as Ockham's Razor aside) in closed systems, *only* good models fit the data ($G \asymp F$); in open systems, the

observed data could have been affected by external influences outside the system. When fitting functions to data from complex open systems (such as social and ecological systems) the ability to exclude or control for external influences is highly constrained. A model of a subsystem that just fits to data will likely also be fitting to external influences on that subsystem.

If a model somehow captures the effect of an external influence that it is not supposed to model, we should be rather suspicious. Further, as Filatova et al. (2016) point out, disturbances to a complex socio-ecological system need not only arise from exogenous influences, but can also grow from endogenous gradual change. If there are multiple 'attractors' and the data have followed one path at a bifurcation but a model follows another, the model will fail to validate. Over multiple runs of the model, of course, it might take the same path as the data did half the time. Given the choice between two models, one of which is simpler, and always follows the path the data did (because it is high-bias, and doesn't bifurcate), and another of which is more complicated, and only follows the path the data did half the time, Ockham's Razor and fit-to-data heuristics tell us to choose the former. However, it is arguably the latter model that has more faithfully captured the underlying dynamics of the system.

The probability of following one trajectory rather than another need not necessarily be 0.5. It could be 1E-6 and it just so happened that this time, the real world followed the one-in-a-million chance trajectory. The model that captures the bifurcation may not be run enough times that the path the data took is observed. The point remains that in complex systems, fit-to-data is not necessarily an indicator that we have a 'good' model. If our model is ontology-free, then it is doubly awful: an oversimplified bendy sheet that hardly reflects the system it is modelling: "It is a tale told by an idiot, full of sound and fury, signifying nothing."[2]

To summarize, validation by fit-to-data is not necessarily (on its own) a helpful measure in complex systems. No matter what the outcome, there exists an argument both for and against the model (Table 1). Nevertheless, it is still potentially useful information about a model, and we show in the box various methods for computing validation error on a set of data, or otherwise comparing models' expected prediction ability. As is apparent from reading Brewer et al. (2016), there is controversy in some of the modelling literature about which measure of expected prediction ability is 'best'. This can lead to reviewers complaining that one measure should have been used rather than another, but since reviewers' statistical fetishes are impossible to predict, we cannot provide further guidance.

[2] Macbeth, Act V, Scene V.

Table 1. Arguments about validation by fit-to-data and whether the model is 'good' or 'bad'

Validation result	Good model	Bad model
Acceptable	The model has fit the data and we estimate it will predict accurately in future.	Although the model has fit to data, it is oversimplified, relies on unrealistic assumptions, doesn't really explain anything, or doesn't allow for the possibility that things could have turned out differently. Its predictions should not be trusted.
Not acceptable	The particular course that history took was highly contingent on phenomena that it would not be reasonable to include in any model. There is a 'possible world' in which the model would be right. Alternatively, the model reproduces 'patterns' (as per Grimm et al. 1996) in the data, if not the data itself. It might still be worth considering the model's predictions.	The model did not fit the empirical data we have, so it must be rejected and its predictions ignored.

Box about here

Box: Metrics of and methods for validation

This box explains various metrics and measures of validation, showing you where to find out more information on them, and how to use them with R. For those of you unfamiliar with R, it is a popularly-used[3] free (as in open-source and in the financial sense) statistical software package, available for Windows, OS-X and Linux.[4] Each of the examples assumes you are validating against a single variable (unless otherwise stated) for which you have a number of samples from your data and corresponding output from your model. The R variable `vdata` contains the empirical data to validate against (which must not have been used for calibration – though many of the metrics can of course be applied to the calibration process); whilst the variable `model` contains the corresponding output from the model. The two variables `vdata` and `model` are, in R terms, vectors of equal length. If the model predicted the data perfectly, then for each element *i* of the two vectors, `vdata[`i`] == model[`i`]`. More information on each of the approaches can be found on Wikipedia,[5] R documentation, and in various machine learning and advanced statistical textbooks.

Metric	**Description**	**R code**
`vdata[`i`], model[`i`]` $\in \mathbb{R}$ (cardinal variable)		

L_1 norm	Total absolute distance between the data and model vector elements. Zero implies a perfect fit.	`sum(abs(vdata - model))`
		
		or
		
		`dm <- rbind(vdata, model)`
		`dist(dm, method = "maximum")`

L_2 norm	The Euclidean distance between the data and model vectors. Zero implies a perfect fit.	`sqrt(sum((vdata - model)^2))`
		
		or
		
		`dm <- rbind(vdata, model)`
		`dist(dm, method = "euclidean")`

[3] Its popularity in the social simulation community is reflected by the fact that tools have been built to link it with Wilensky's (1999) Netlogo (Thiele et al. 2012).

[4] http://www.r-project.org/ <Accessed May 2017>

[5] https://www.wikipedia.org/ <Accessed May 2017>

p norm	The *p*th root of the sum of the *p*th power of the differences between the data and model vector elements. Zero implies a perfect fit.	`dm <- rbind(vdata, model)` `dist(dm, method = "minkowski", p = p)`

L_∞ norm	The maximum difference between any pair of values. Zero implies a perfect fit.	`max(abs(vdata - model))`
		
		or
		
		`dm <- rbind(vdata, model)`
		`dist(dm, method = "maximum")`

Sum of squared error (SSE)	Sum of squared error is simply the square of the L_2 norm. Zero implies a perfect fit.	`sum((vdata - model)^2)`
RMS error	Root mean squared error is the L_2 norm corrected for the size of the data. Zero implies a perfect fit.	`sqrt(sum((vdata - model)^2) / length(vdata))`
`vdata[`*i*`]`, `model[`*i*`]` ∈ {0, 1} (Boolean variable) This case could also be used for nominal variables (i.e. classes, with one number being used for each class), if repeated for each class. Here, for each class *j*: `vdata[`*i*`] <- ifelse(vdata[`*i*`] ==` *j*`, 1, 0)` and `model[`*i*`] <- ifelse(model[`*i*`] ==` *j*`, 1, 0)`.		
Precision or Positive Predictive Value (PPV)	The precision is the cases where the model correctly said a phenomenon occurred as a fraction of all the cases where the model said the phenomenon occurred. Ideally it would be 1.	`sum(ifelse(vdata == 1 & model == 1, 1, 0))` `  / sum(ifelse(model == 1, 1, 0))`

Recall, Sensitivity, or True Positive Rate (TPR)	The recall is the cases where the model correctly said a phenomenon occurred as a fraction of all the cases where the data said the phenomenon occurred. Ideally it would be 1.	`sum(ifelse(vdata == 1 & model == 1, 1, 0))` `  / sum(ifelse(vdata == 1, 1, 0))`

F measure
 	
or
 	
F_1 score
	The F measure or F_1 score is the harmonic mean of precision and recall, scaled such that it is 1 in the ideal case and 0 in the worst case.	`p <- sum(ifelse(vdata == 1 & model == 1, 1, 0))`
		`  / sum(ifelse(model == 1, 1, 0))`
		
		`r <- sum(ifelse(vdata == 1 & model == 1, 1, 0))`
		`  / sum(ifelse(vdata == 1, 1, 0))`
		
		`2 * (t * p) / (t + p)`

False Positive Rate (FPR) or fall-out	The false positive rate is the equivalent of Type I error – and is the cases where the model has predicted an occurrence of the phenomenon as a fraction of the number of cases where it doesn't in the data.	`sum(ifelse(vdata == 0 & model == 1, 1, 0))` `  / sum(ifelse(vdata == 0, 1, 0))`
False Negative Rate (FNR) or miss rate	The false negative rate is the equivalent of Type II error – the cases where the model has predicted the phenomenon does not occur as a fraction of the number of cases where it does in the data.	`sum(ifelse(vdata == 1 & model == 0, 1, 0))` `  / sum(ifelse(vdata == 1, 1, 0))`

Cohen's kappa	Cohen's kappa is a statistic that measures the degree of agreement	`require(vcd)`
		`m <- matrix(c(sum(ifelse(vdata == 1 & model == 1, 1, 0),`
		`  sum(ifelse(vdata == 0 & model == 1, 1, 0),`

	between the data and the model by comparing the observed agreement with an estimate of the probability of agreement occurring by chance.	`sum(ifelse(vdata == 1 & model == 0, 1, 0),`
		`sum(ifelse(vdata == 0 & model == 0, 1, 0)),`
		`nrow = 2, ncol = 2)`
		`Kappa(m)`

Various metrics are available for the case where the model is stochastic and has a distribution of outputs for a given parameter setting. In this case, `model` is now a matrix with the same number of rows as the length of `vdata`, and one column for each replication of the model with the parameter setting being evaluated, and inputs corresponding to the row in `vdata`.		

Likelihood	Likelihoods are typically computed using probability distributions, and in R, are available as computations for specific model-fitting algorithms (such as linear models). Essentially, the likelihood is a function of the parameters of a model given the observed data. For a given set of parameter values, a higher likelihood suggests a better fit to the data. Calibration could search for maximum likelihood parameter values. Unless the distribution of the model's prediction is known (as a function), likelihoods can only be estimated. Log likelihoods are sometimes reported as they can be easier to compute in the case of known distributions.	First we build a matrix `meq` to store the cases where the model has produced the same output as the data:
		
		`meq <- apply(model, 2, `==`, vdata)`
		
		Strict equality between the model and the data is typically unrealistic; `meq` would then normally be full of `FALSE` entries. In mathematical models, the Gaussian 'noise' term usually caters for this issue, but requires *k* in the AIC and BIC to be incremented accordingly. An alternative, allowing an accuracy, `epsilon` (which similarly counts as another parameter):
		
		`near.enough <- function(x, y, epsilon = 0.01) {`
		`  abs(x - y) < epsilon`
		`}`
		`meq <- apply(model, 2, near.enough, vdata)`
		
		Continuing with the computation of likelihood, `ceq` is a vector (with one element for each column in `model`) saying whether *all* the model's outputs are equal (or near enough) to `vdata`. This can then be used to compute an estimated probability of the model producing acceptable output.
		
		`ceq <- apply(meq, 2, all)`
		`est.likelihood <- sum(ifelse(ceq, 1, 0)) / length(ceq)`

Akaike	The AIC is an information theoretic	Although R provides a function to compute the AIC (the aptly named `AIC()`), this

Information Criterion (AIC)	measure that can be used for comparing various models of the same data. The models must have been calibrated by selecting the maximum likelihood parameterization – a restriction than makes the application to ABM challenging. The absolute value of the AIC is not of any interest outside a model comparison context.
	
	The AIC is intended to represent the information loss associated with using the model to estimate the data, and includes a penalization of the number of parameters the model has (*k*). For our purposes, *k* is minimally the number of parameters you have been adjusting in the model during calibration. If your model has samples from probability distributions with hard-coded parameters, *k* should be incremented for each of those parameters too. (This would be equivalent to incrementing *k* for mathematical models with a
		function uses the likelihood, and hence assumes it is being passed a fitted model object, such as a linear model. For ABMs, the AIC has to be computed (estimated) by hand using the estimated likelihood. Here we assume we have two models with output matrices `model1` and `model2`, numbers of parameters `k1` and `k2`, and estimated likelihoods `est.lik1` and `est.lik2`, both of which have been computed using the same `vdata`.
		
		`est.aic1 <- 2 * k1 - 2 * log(est.lik1)`
		`est.aic2 <- 2 * k2 - 2 * log(est.lik2)`
		
		If `est.aic1 < est.aic2`, then `model2` is `exp((est.aic1 - est.aic2) / 2)` times as probable as `model1` to minimize the information loss with respect to the data. If the two models are 'reasonably close' (e.g. `exp((est.aic1 - est.aic2) / 2) > 0.5`), and it makes sense to do so, their predictions can be weighted: `model1` by 1, `model2` by `exp((est.aic1 - est.aic2) / 2)`; otherwise, the model with higher AIC would be rejected.

	Gaussian 'noise' parameter.) Debate about what might 'count' in your ABM's program code for the purposes of incrementing *k* could well form the basis of further questioning the applicability of the AIC to ABMs. Models with smaller AIC are preferred.	
Bayes Information Criterion (BIC)	The BIC is based on Bayesian principles for model selection, and uses a stronger penalty term for parameters than the AIC: $k \log n$, where *k* is the number of parameters (as in the AIC, and hence with the same issues when applying to ABM), and *n* is the length of the `vdata` vector. It is required that $n >> k$. Just as with the AIC, models with smaller BIC are preferred.	`est.bic <- k * log(length(vdata)) - 2 * log(est.likelihood)`

Validating ontologies

After summarizing the foregoing arguments, this section elaborates more on the structure of the model, which may be referred to as its 'ontology'. After briefly introducing ontologies, we build an argument for why agent-based models have the scope to pay more attention to this side of modelling based on the *expressivity* of a formal language for writing ontologies. We then consider various ways in which ontologies could be 'validated' – in the sense of establishing confidence in them, finding that this is far from being a sufficiently settled area that a box could be provided such as the one we have provided for fit-to-data.

The foregoing pages had two objectives. One was to summarize all the different ways people try to estimate how well their model has fit some empirical data, to give them some kind of (preferably quantitative) idea of how much they should believe in its predictions. The other is to argue that there's more to evaluating a model than just looking at its fit-to-data, largely by showing various ways in which fit-to-data may not be as convincing an indicator of a model's suitability as some appear to believe it to be. To summarize the reasons, the first two of which may seem a little 'unfair', but should be anticipated in complex social systems:

- Simplifying assumptions that apply during calibration may not apply at prediction.
- The (formal) language you have used to represent the system during calibration may not be adequate during prediction.
- You may not have enough data to justify a model with a high VC dimension; but using a model with a lower VC dimension would be oversimplifying.
- In complex/non-ergodic systems, at a bifurcation point, the empirical data may have followed a path that had a low probability in comparison with other paths it could have taken.

The various methods for measuring estimated prediction ability say relatively little about the structure of the model itself, except, in the case of metrics like the AIC and BIC, by penalizing models for having too many parameters. In neural networks, this is the number of weights the network has, but assumptions about functional form are embedded in the structure of the network itself – how the nodes are arranged into layers and/or connected to each other. This structure, however, only reflects the flexibility the network will have to achieve certain combinations of outputs on all the inputs it might be given (its 'wiggliness'). This is a rather weak ontological commitment to make to a set of data.

Neural networks are an extreme – one in which there is the minimum representative connection between the empirical world and the nodes and network of connecting weights that determine the behaviour of the model. They are nevertheless useful when there is a large amount of data available for training, the modelled system isn't complex, and one is not particularly concerned about how the input-output mapping is achieved; only that whatever mapping obtained has good prediction ability.

Neural networks are very interesting to contrast with agent-based models, which also feature networks of behaving entities, but where the network of connections and the behaving entities are supposed to have a representative link with the empirical world. In the artificial intelligence community, this representative structure would be referred to as the *microworld* (e.g. Chenoweth 1991) of the simulation. A famous example is Winograd's (1972) Blocks World. However, with advances in formal languages for expressing such representative structure, we could also refer to these microworlds as ontologies.

Ontologies in computer science are defined by Gruber (1993) as formal, explicit representations of shared conceptualizations. In general, ontologies cover a broad range of formalized representations, including diagrams, computing code, even the structure of a filesystem, but the development of description logics (Baader and Nutt 2003) means that there are formal languages for ontologies to which automated reasoning can be applied. One of the most popularly-used languages for ontologies, which draws on description logics, is the Web Ontology Language (OWL; Cuenca Grau et al. 2008; Horrocks et al. 2003). The application of OWL to agent-based modelling has been discussed by a number of authors (e.g. Gotts and Polhill 2009; Livet et al. 2010), but of particular relevance for our purposes is the application of OWL to representing the structure of agent-based models (Polhill and Gotts 2009) and any mappings between the programming languages used for implementing agent-based models and OWL ontologies (Polhill 2015; Troitzsch 2015).

For the purposes of highlighting why the ontology of an agent-based model becomes so much more significant, one of the measures of a description logic is its expressivity. The expressivity of a logic is essentially the various kinds and combinations of axiom it allows you to create, whilst still having decidable reasoning. We might compare different modelling approaches according to the ontological expressivity needed to capture descriptions of the states the model can have. Description logics use a letter-based notation to describe the axioms each logic has (Baader and Nutt 2003; Calvanese and De Giacomo 2003; Baader et al. 2003). Briefly, $\mathcal{AL}$ is a basic description logic, and $^{(\mathcal{D})}$ is for data properties; $\mathcal{C}$ provides more complex class axioms than the basic axioms in $\mathcal{AL}$; $\mathcal{R}$ is for complex relationship assertions such as irreflexivity (all NetLogo links are irreflexive, for example, as you cannot link anything to itself); $\mathcal{O}$ introduces nominals (a bit like enumerations in Java); $\mathcal{I}$ inverse relationships; $\mathcal{N}$ numerical restrictions on properties; and $\mathcal{F}$ functional properties. Table 2 provides an initial indication of the description logic expressivity needed to capture the syntax used to specify the ontologies of various modelling approaches. However, the labels applied in the 'description logic' column do not necessarily mean that the full capabilities of the language are necessarily used.

The fact that agent-based models have a generally richer expressivity for defining the ontologies over which they operate means that some of the complaints of qualitative social researchers about quantitative social researchers are brought into sharper focus. The ontology of an agent-based model is less constrained by the amount of data available, aesthetic concerns about elegance,

or the need to reduce the number of variables to enable tractable mathematical evaluation of equilibria.

Table 2. Comparison of expressivity of ontologies of various modelling approaches. Letters are used to represent terms, or groups of terms needed to capture any syntax for the modelling approach's formalism with respect to the real world. See text for an explanation.

Modelling approach	**Expressions needed**	**Description logic**	**Comments**
Neural networks	The concepts of inputs and outputs, and data property labels for each node.	$\mathcal{AL}^{(\mathcal{D})}$	The only ontologically significant terms are the input and output variables. Rudimentary classes are needed for Input and Output specifically.
ODEs	Data properties for each variable, distinction between exogenous and endogenous variables, causal influence.	$\mathcal{ALCOIN}^{(\mathcal{D})}$	Concepts would be needed for each variable so that causal influences can be represented with relationships.
System dynamics	As ODEs, but stocks and flows are also relevant concepts.	$\mathcal{ALCOIN}^{(\mathcal{D})}$	Stocks and flows as concepts do not add any extra requirements for expressivity.
Social Network Analysis	Individuals and relationships, possibly data properties where attributes of individuals relevant.	$\mathcal{ALI}^{(\mathcal{D})}$	Concepts not really needed (other than Top) so $\mathcal{ALI}$ is more expressive than SNA really requires. Data properties optional.
Agent-based modelling	Classes, inheritance, individuals, data properties, object properties, lists, arrays, domain and ranges needed.	$\mathcal{ALCROINF}^{(\mathcal{D})}$	Not all agent-based models will need all the expressivity options. If you have a NetLogo model and want to find out the expressivity of its ontology, you can use Polhill's (2015) automated ontology extraction tool, load the result into Protégé, and the ontology summary tab tells you the description logic needed. For example, Ge and Polhill's (2016) model of commuting has description logic $\mathcal{ALRIF}^{(\mathcal{D})}$.

It is also much clearer that the ontology is by-and-large a subjective choice. Nevertheless, we wish to have an idea of how 'good' that subjective choice is – something that may be as much about normativity in the community with an interest in the model as (supposedly) objective numerical measures. That said, if we are to move beyond fit-to-data as the sole basis for our belief in the predictions of a model, we still need some ways of assessing the model's ontology as an additional basis for such belief. This is far from being in a position where there are established methods, but four ways in which an ontology can be assessed are:

- logical consistency;
- populating it with instances;
- stakeholder and/or expert evaluation; and
- comparison with existing ontologies.

If the ontology can be translated into OWL, the first of these of these can be achieved using the consistency checking available in reasoning applications such as such as Pellet (Sirin et al. 2007), FACT++ (Tsarkov and Horrocks 2006), HermiT (Shearer et al. 2008) and Ontop (Bagosi et al. 2014). Though consistency checking ensures we have at least made no logical contradictions in our specification of the ontology, it is rather a low bar to set as it says little about the quality of the representation. Beyond mere logical consistency, there are methodologies such as OntoClean (Guarino and Welty 2009) for validating the ontological adequacy of taxonomies. However, this also says more about the correctness of the operational semantics of a given set of axioms in an ontology, as opposed to addressing the sufficiency of that ontology to represent a given problem domain.

Populating an ontology with instances is another check of the 'validity' of an ontology, as difficulties with so doing, especially with empirical data, can reveal where the ontology is 'awkward' in its specifications. Working ontologies are produced every time a successful IT project is implemented. Any modern enterprise system is usually the result of a problem domain being modelled using some object-oriented analysis and design (Rumbaugh 2003) and as such necessarily involves visual modelling (usually Unified Modelling Language – UML). The resultant conception is then implemented in one of the numerous object-oriented computer languages. Although not formally provable as in any way equivalent, such systems are *prima facie* evidence of the success construction of working ontologies, albeit normally in UML. Although not equivalent, design practices can be implemented that result in a one-to-one translation between UML and OWL (Object Modelling Group 2014, p. 130). Embedded software systems operating machinery in the real world (e.g. autopilots and control systems) have their ontologies validated every time they send a signal to a servo or relay, which over time constitutes a robust empirical test of their conceptualizations. From an agent-based modelling perspective, where the ontology describes the entities and state variables in the model, pragmatic issues with the ontology could become apparent when trying to populate the model from empirical databases. However, since the schemas of these databases are themselves ontologies, there is the potential to argue that it is those ontologies, or the integration thereof, that is the locus of any problems, rather than with the model itself. Hence, unless the context is embedded software, the ability to initialize a model from empirical data is also a rather weak test of the validity of the model's ontology.

The third idea of stakeholder and/or expert evaluation involves a degree of integration of specific problem-domain knowledge and ontological engineering expertise if we are to be convinced that the evaluators have really understood the implications of the formalization of their knowledge. Sowa (1999, p. 452) points out that knowledge engineering is a specialism requiring skills in logic, language and philosophy that domain experts should not be expected to have. Even if experts agree on a conceptualisation of a domain, they will not necessarily be able to construct ontologies of it; this will be done instead by the knowledge engineer. The resulting ontology is the knowledge engineer's

conceptualization of the experts' conceptualization, and may differ from one knowledge engineer to another. Such problems and in particular their relevance to the veridicality and the actual information content of natural language utterances such as those from domain experts is extensively discussed by Devlin (1991, ch. 1-2).

There are formal methodologies available for knowledge elicitation, such as On-To-Knowledge (Sure et al. 2004); creating ontologies from existing thesauruses, or taxonomies, as illustrated by Huhn and Schulz (2004); and those listed by Jones et al. (1998). However, such methodologies would normally be associated with model design rather than model validation. Since validation is only really meaningful when using 'out-of-sample' data (i.e. data not used for calibration), we should expect validation of model ontologies to be a process that behaves equivalently, for example, through using different experts during validation than during model design. In the case of peer-reviewed journal articles, this arguably happens automatically assuming that reviewers have had nothing to do with the work. However, validation by peer-review detracts from the sense of reporting on a completed piece of work in a journal article, and is not something that is typically documented, except in more innovative open access journals such as Earth System Dynamics,[6] where reviews and authors' responses are also available to read. Whether validating with academic peers or with non-academic participants or stakeholders in a model, issues with the conceptualization highlighted during validation may reflect controversies and differences in conceptualization in the community rather than issues with the particular conceptualization in the model as such.

Using formal knowledge elicitation methods, such as those listed above, to build new ontologies from the experts involved in model validation rather than those involved in model design may seem excessive. Polhill et al. (2010) document a process by which assumptions in the formalization are converted back to natural language and then 'checked' (they use this somewhat weaker term than 'validation' to describe the process) with domain experts. Since expert validation is, formally or informally, essentially a process of ontology comparison, a rigorous approach to validating ontologies would involve two knowledge elicitation exercises – one during design, and one during validation.

Ontology comparison can be seen as matching ontological primitives between at least two differing ontologies. In the world of ontologies, however, such linking of primitives between ontologies is referred to as *interoperability*. Interoperability refers to the conditions under which we can establish a formal correspondence between two ontological primitives. Though interoperability was a motivation for the development of the semantic web (Berners-Lee et al. 2001), interoperability between ontologies has been somewhat intractable historically (Kalfoglou and Schorlemmer 2003), and indeed may have stalled the widespread adoption of ontologies in other application domains.

---

[6] http://www.earth-system-dynamics.net/ <Accessed May 2017>

Pragmatically, interoperability is hampered by issues that come under the heading of *semantic heterogeneity*, in which there are various semantic conflicts (see, e.g. Bellatreche et al. 2006), from the seemingly trivial naming conflicts (the same name for different concepts, or different names for the same concepts) to the more significant representation conflicts (concepts are represented in the different ways). However, there are also deeper philosophical issues to do with whether ontologies are seen as being 'observed' or 'constructed' (see Klein and Hirschheim 1987). If the former, then we should expect to find commonality in conceptualizations because we all see the same world and discriminate the same entities in it; if the latter, such commonality is a function of norms in the way the external world is conceptualized, and any differences are cultural (and hence subject to political connotations if one conceptualization is argued to be 'better' than another). Grubic and Fan (2010, p. 783), reviewing ontologies of supply chains, conclude by noting the need to challenge the perception that building ontologies is simply a problem of terminology – finding the 'right' names for things in the real world.

With all the above caveats in mind, there are a few approaches to ontology interoperability, with some tools listed in Table 3:

- Token matching, or token transformation – this makes use of automated token matching via textual analysis or leveraging existing ontologies to provide correspondences between previously unrelated ontological entities. Most of the tools in Table 3 use this kind of matching at some level.
- Graph analysis of the ontology. This includes Formal Concept Analysis (FCA), which uses graphs to link informationally related items (Yang and Feng 2012), and other general graph matching or analysis algorithms such as in S-Match (Giunchiglia et al. 2012).
- Machine learning – Examples of this include GLUE (Doan et al. 2004) and the more recent YAM++ (Ngo and Bellahsene 2012), both using machine learning to try and create correspondences between ontological elements.
- Information Flow (IF) or semantic information content – this has been around quite a while, but is still largely theoretical (Barwise and Seligman 1997). Premised by the externalist assumption and the assumption of veridical nature of information, this is treatment of information and its relations using category theory (Kalfoglou and Schorlemmer 2003). There are no useful implementations of this methodology so far.
- Some combination of all the above.

Token matching and graph analysis are most prevalent. An additional review of the available systems and software for ontology interoperability can be found in Shvaiko and Euzenat (2013), and some other, older, but still useful methodologies may also be found in Jean-Mary et al. (2009).

Potentially, therefore, the tools and infrastructure exist to evaluate interoperability between domain and model ontologies. The 'validation data' would comprise a pre-existing domain ontology not used to build the model, or a domain ontology obtained through a second knowledge elicitation exercise with

experts or stakeholders. The model's ontology could be extracted automatically (e.g. using tools such as Polhill's (2015) NetLogo extension, or appropriately designed object-oriented programs enabling exploitation of one-to-one mappings from UML to OWL) or manually, and then applications such as those in Table 3 used to assess their interoperability. Such an exercise is rather more effort than fit-to-data validation: the maturity of the area is far from being in a position where it is simply a matter of invoking a function call in the appropriate R library as in the examples in the Box.

Table 3 Available ontology interoperability tools, method used for interoperability, and licence. The list is based on work by Bergman (2014). The tools are all implemented in Java.

Tool	**Brief description**	**Method**	**Licence**
AgreementMakerLight	An automated and efficient ontology matching system derived from AgreementMaker (Faria et al. 2013)	Matching	Apache
COMA++	A schema and ontology matching tool with a comprehensive infrastructure. Its graphical interface supports a variety of interaction (Do and Rahm 2002).	Matching	AGPL
Falcon-AO (Finding, aligning and learning ontologies)	This is an automatic ontology matching tool that includes the three elementary matchers of string, virtual documents and graph similarity measures. In addition, it integrates a PBM (Partition-based Block Matching; Hu et al. 2008) algorithm to cope with large-scale ontologies (Hu and Qu 2008).	Matching	Open Source
OnAGUI (Ontology Alignment Graphical User Interface)	This is an alignment helper and viewer that also makes automatic discovery of alignment using different kind of algorithms.	Manual, Graph	GPL
S-Match	Takes any two tree like structures (such as database schemas, classifications, lightweight ontologies) and returns a set of correspondences between those tree nodes which semantically correspond to one another (Giunchiglia et al. 2012).	Graph	LGPL
YAM++ (Yet Another Matcher)	A self-configuring ontology matching system for discovering semantic correspondences between entities (i.e., classes, object properties and data properties) of ontologies using machine learning (Ngo and Bellahsene 2012).	Machine Learning	Open Source

There is also the issue that effectively the model is assessed twice, once with respect to its fit-to-data (which is still information, even if arguably not dependable as a sole indicator of how 'good' a model is), and once with respect to its ontology. If we are not to assume that a richer ontology automatically leads to a better fit to data, the trade-off between fit-to-data and ontological interoperability is not a trivial choice to make. Even in more established model assessment metrics that use some information about model structure, the differences in penalty of parameters between the AIC and BIC illustrates the scope for potential controversy. Indeed, Brewer et al. (2016) argue that the choice of which of these to use is sensitive to context, suggesting there is no

universally applicable trade-off heuristic. In the meantime, in the social simulation community, common practice is either to use stakeholder evaluation in participatory contexts, or simply to rely on peer-review. Given that ontological expressivity is a major advantage for agent-based modelling, at least as suggested by our categorizations in Table 2, the community should set itself the aim of finding ways to quantify that benefit.

We emphasize that our arguments do not mean that fit-to-data can be ignored as a criterion for assessing the confidence we should have. Rather, that we *also* need to pay attention to the model's ontology. As this section has shown, there are various ways to do this, though the area is far from being sufficiently settled that we can provide 'generally used' quantitative measures of the fit of an ontology to a system. This may reflect the fact that agent-based simulation, a relatively recent development in the world of modelling, has a much greater potential expressivity in its ability to specify ontologies, and the question of model structure has thus far been limited to discussions about numbers of parameters. Further, there is evidence that we should not expect to find a single, general measure that appropriately trades off fit-to-data and ontological fit and provide us with a number that tells us how 'good' a model is.

Conclusion

Summarizing the key arguments in this chapter:

- Methods for validating models have thus far concentrated on fit-to-data, and there are various ways in which that fit can be assessed.
- However, fit-to-data, though it should not be ignored, cannot be trusted as the sole basis for model validation. Besides questions about comparability of context, the modeller's biases in encoding, or representing, the system need to be questioned. Further, in complex open systems, fit-to-data does not resolve whether a model's predictions should be trusted.
- Agent-based models have greater potential ontological expressivity than other modelling approaches, and researchers wishing to validate their models need to pay attention to their ontology as well as their fit-to-data.

The effort involved in building an agent-based model in an empirical context, as opposed to a more traditional aggregate-level mathematical model, is predicated on the empirical world being 'complex'. In such systems, validation by fit-to-data is not, on its own, a sound basis for estimating the ability of a model to make reliable predictions, not least because of issues with path dependency. However, the availability of sufficient data to justify building a model with as many parameters as an agent-based model typically has is a further significant potential issue, at least until methods are developed to assess how flexible agent-based models are in their ability to realize input-output mappings. These points, however, apply just as much to any other modelling approach as they do to agent-based models, noting that models with a number of parameters commensurate with the available data may be oversimplifying.

Unsatisfactory though some will find the idea that a model's ontology is subjective, the increased expressivity of agent-based models' ontologies over

those of other formal modelling approaches places greater onus on the assessment of these ontologies as part of the validation process. Methodologies for assessing ontologies are still not at a sufficiently mature stage that there is a clear 'standard', though we have argued that best practice would involve a separate knowledge elicitation exercise with experts not involved in design, and a comparison of the resulting ontology (or an ontology generated from a similar process in by other authors) with that of the model. Given interest in ontologies in other disciplines, there is an opportunity for the agent-based modelling community to contribute to this area, ensuring that tools and techniques can be tailored to meet any specific requirements.

Acknowledgements

We acknowledge funding from the Engineering and Physical Sciences Research Council (award no. 91310127), the European Commission Framework Programme 7 'GLAMURS' project (grant agreement no. 613420) and the Scottish Government Rural Affairs, Food and the Environment Strategic Research Programme, Theme 2: Productive and Sustainable Land Management and Rural Economies. We are also grateful to Bruce Edmonds and Mark Brewer for useful comments on earlier drafts of this chapter; any mistakes are of course our own.

Further reading

- Shalizi's (2006) book chapter covers approaches to modelling (and measuring) complex systems in a more formal and comprehensive way, with a focus on more traditional mathematical modelling techniques. However, he also covers issues with validation and penalization of parameters, including discussions of VC theory and Ockham's razor.
- Sowa's (1999) book on knowledge representation is a good introduction to various issues in the field, and covers various formalisms and underlying philosophical questions that the formal representation of knowledge yields. Baader et al.'s (2003) *Description Logic Handbook* goes in to more details on description logics. Another book, which goes into some depth on controversies in the formal representation of what otherwise seems to be a simple everyday concept, 'if-then', is Evans and Over's (2004) book, and this too is highly recommended.
- Since one of the ways of validating ontologies is through engaging with stakeholders, the Companion Modelling school of agent-based modelling, pioneered especially by research teams based in France, is well worth familiarizing yourself with. They have a website,[7] and a book (Etienne 2014) as well as several publications illustrating their work. Since they sometimes use ontologies as part of their methodological approach to modelling with stakeholders, the work of authors such as Jean-Pierre Müller, Nicolas Becu and Pascal Perez and their collaborators are particularly worth investigating. Some example articles include Müller (2010), Becu et al. (2003) and Perez et al. (2009). Companion modellers are not the only ones to apply knowledge elicitation to model design, however – see, for example, Bharwani et al. (2015).

[7] https://www.commod.org/en <Accessed May 2017>

- Validation has long been a subject of discussion in agent-based modelling, and this chapter has not dedicated space to reviewing the excellent thinking that has already been done on the topic. The interested reader wanting to access some of this literature is advised to look for keywords such as *validation*, *calibration*, and *verification* in the *Journal of Artificial Societies and Social Simulation*, currently the principal journal for publication of agent-based social simulation work. Notable recent articles include Schulze et al. (2017), Drchal et al. (2016), ten Broeke et al. (2016), and Lovelace et al. (2015). Other older articles worth a read are Elsenbroich (2012), Radax and Rengs (2010) and Rossiter et al. (2010). See also some of the debates such as Thompson and Derr's (2009) critique of Epstein's (2008) article and Troitzsch's (2009) response, and Moss's (2008) reflections on Windrum et al.'s (2007) paper. A practical article on one approach to validating agent-based models outwith *JASSS* is Moss and Edmonds (2005).

References

Abu-Mostafa, Y. S. (1989) The Vapnik-Chervonenkis dimension: Information versus complexity in learning. *Neural Computation* **1** (3), 312-317.

Aha, D. W. (1992) Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies* **36** (2), 267-287.

Baader, F. and Nutt, W. (2003) Basic description logics. In Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. and Patel-Schneider, P. F. (eds.) *The Description Logic Handbook*. New York, NY, USA: Cambridge University Press, pp. 43-95.

Baader, F., Küsters, R. and Wolter, F. (2003) Extensions to description logics. In Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. and Patel-Schneider, P. F. (eds.) *The Description Logic Handbook*. New York, NY, USA: Cambridge University Press, pp. 219-261.

Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., Rodriguez-Muro, M., Slusnys, M. and Xiao, G. (2014) The Ontop framework for ontology based data access. In Zhao, D., Du, J., Wang, H., Wang, P., Donghong, J. and Pan, J. Z. (eds). *The Semantic Web and Web Science. 8th Chinese Conference, CSWS 2014, Wuhan, China, August 8-12, 2014, Revised Selected Papers*. Berlin, Germany: Springer-Verlag, pp. 67-77.

Barwise, J. and Seligman, J. (1997) *Information Flow: The Logic of Distributed Systems*. Cambridge, UK: Cambridge University Press.

Bellatreche, L., Xuan Dong, N., Peirra, G. and Hondjack, D. (2006) Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry* **57**, 711-724.

Becu, N., Bousquet, F., Barreteau, O., Perez, P. and Walker, A. (2003) A methodology for eliciting and modelling stakeholders' representations with

agent based modelling. In Hales, D., Edmonds, B., Norling, E. and Rouchier, J. (eds.) *Multi-Agent-Based Simulation III. MABS 2003. Lecture Notes in Computer Science* **2927**. Berlin, Heidelberg: Springer, pp. 131-148.

Bergman, M. (2014) *50 ontology mapping and alignment tools*. http://www.mkbergman.com/1769/50-ontology-mapping-and-alignment-tools/ <Accessed May 2017>

Berners-Lee, T., Hendler, J. and Lassila, O. (2001) The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American* **284** (5), 28-37.

Bharwani, S., Besa, M. C., Taylor, R., Fischer, M., Devisscher, T. and Kenfack, C. (2015) Identifying salient drivers of livelihood decision-making in the forest communities of Cameroon: Adding value to social simulation models. *Journal of Artificial Societies and Social Simulation* **18** (1), 3. http://jasss.soc.surrey.ac.uk/18/1/3.html <Accessed May 2017>

Bishop, C. M. (1995) *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press.

Brewer, M. J., Butler, A. and Cooksley, S. (2016) The relative performance of AIC, AIC_C and BIC in the presence of unobserved heterogeneity. *Methods in Ecology and Evolution* **7**, 679-692.

Calvanese, D. and De Giacomo, G. (2003) Expressive description logics. In Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. and Patel-Schneider, P. F. (eds.) *The Description Logic Handbook*. New York, NY, USA: Cambridge University Press, pp. 178-218.

Cheng, B. and Titterington, D. M. (1994) Neural networks: A review from a statistical perspective. *Statistical Science* **9** (1), 2-30.

Chenoweth, S. V. (1991) On the NP-Hardness of blocks world. *AAAI-91 Proceedings*, pp. 623-628.

Chester, D. L. (1990) Why two hidden layers are better than one. *Proceedings of the International Joint Conference on Neural Networks, 15 19 January 1990 Washington DC,* Vol. 1, pp. 265-268.

Clarke, K. A. (2005) The phantom menace: Omitted variable bias in econometric research. *Conflict Management and Peace Science* **22** (4), 341-352.

Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P. and Sattler, U. (2008) OWL 2: The next step for OWL. *Journal of Web Semantics* **6** (4), 309-322.

Cybenko, G. (1989) Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems* **2** (4), 303-314.

Devlin, K. (1991) *Logic and Information*. Cambridge, UK: Cambridge University Press.

Do, H.-H. and Rahm, E. (2002) COMA: A system for flexible combination of schema matching approaches. *VLDB 2002: 28th International Conference on Very Large Data Bases, Kowloon Shangri-La Hotel, August 20-23, 2002, Hong Kong, China*. http://www.vldb.org/conf/2002/S17P03.pdf <Accessed May 2017>

Doan, A., Madhavan, J., Domingos, P. and Halevy, A. (2004) Ontology matching: A machine learning approach. In Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. Berlin, Germany: Springer-Verlag, pp. 385-403.

Drchal, J., Čertický, M. and Jakob, M. (2016) VALFRAM: Validation framework for activity-based models. *Journal of Artificial Societies and Social Simulation* **19** (3), 15. http://jasss.soc.surrey.ac.uk/19/3/15.html <Accessed May 2017>

Edmonds, B. (2002) Simplicity is not truth-indicative. *Centre for Policy Modelling Discussion Papers CPM-02-99, 3 June 2002*. http://cfpm.org/discussionpapers/111/simplicity-is-not-truth-indicative <Accessed May 2017>

Edmonds, B. and Moss, S. (2005) From KISS to KIDS: An 'anti-simplistic' modelling approach. In Davidsson P., Logan B. and Takadama K. (eds.) *Multi-Agent and Multi-Agent-Based Simulation, Joint Workshop MABS 2004, New York, NY, USA, July 19, 2004, Revised Selected Papers. Lecture Notes in Artificial Intelligence* **3415**, pp. 130-114.

Elsenbroich, C. (2012) Explanation in agent-based modelling: functions, causality or mechanisms? *Journal of Artificial Societies and Social Simulation* **15** (3), 1. http://jasss.soc.surrey.ac.uk/15/3/1.html <Accessed May 2017>

Epstein, J. M. (2008) Why model? *Journal of Artificial Societies and Social Simulation* **11** (4), 12. http://jasss.soc.surrey.ac.uk/11/4/12.html <Accessed May 2017>

Etienne, M. (2014) *Companion Modelling: A Participatory Approach to Support Sustainable Development*. The Netherlands: Springer.

Evans, J. St. B. T. and Over, D. E. (2004) *If*. Oxford, UK: Oxford University Press.

Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I. F. and Couto, F. M. (2013) The AgreementMakerLight ontology matching system. In Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P. and Dou, D. (eds.) *On the Move to Meaningful Internet Systems: OTM 2013 Conferences. Confederated International Conferences CoopIS, DOA-Trusted Cloud, and ODBASE 2013, Graz, Austria, September 9-13, 2013. Proceedings. Lecture Notes in Computer Science* **8185**, pp. 527-541.

Filatova, T., Polhill, J. G. and van Ewijk, S. (2016) Regime shifts in coupled socio-environmental systems: Review of modelling challenges and approaches. *Environmental Modelling and Software* **75**, 333-347.

Funahashi, K. (1989) On the approximate realisation of continuous mappings by neural networks. *Neural Networks* **2** (3), 183-192.

Ge, J. and Polhill, J. G. (2016) Exploring the combined effect of factors influencing commuting patterns and CO2 emissions in Aberdeen using an agent-based model. *Journal of Artificial Societies and Social Simulation* **19** (3), 11. http://jasss.soc.surrey.ac.uk/19/3/11.html <Accessed May 2017>

Giunchiglia, F., Autayeu, A. and Pane, J. (2012) S-match: an open source framework for matching lightweight ontologies. *Semantic Web* **3** (3): 307-317.

Gotts, N. M. and Polhill, J. G. (2009) Narrative scenarios, mediating formalisms, and the agent-based simulation of land use change. In Squazzoni, F. (ed.) *Epistemological Aspects of Computer Simulation in the Social Sciences. Second International Workshop EPOS 2006, Brescia, Italy, October 5-6, 2006. Revised Selected and Invited Papers. Lecture Notes in Artificial Intelligence* **5466**, pp. 99-116.

Gotts, N. M. and Polhill, J. G. (2010) Size matters: large-scape replications of experiments with FEARLUS. *Advances in Complex Systems* **13** (4), 453-467.

Grimm, V., Frank, K., Jeltsch, F., Brandl, R., Uchmański, J. and Wissel, C. (1996) Pattern-oriented modelling in population ecology. *The Science of the Total Environment* **153**, 151-166.

Gruber, T. R. (1993) A translation approach to portable ontology specification. *Knowledge Acquisition* **5** (2), 199-220.

Grubic, T. and Fan. I.-S. (2010) Supply chain ontology: review, analysis and synthesis. *Computers in Industry* **61**, 776-786.

Guarino, N. and Welty, C. A. (2009) An overview of OntoClean. In Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. Berlin: Springer Verlag, pp. 201-220.

Gurney, K. (1997) *An Introduction to Neural Networks*. London, UK: UCL Press.

Hanson, S. J. and Burr, D. J. (1990) What connectionist models learn: learning and representation in connectionist networks. *Behavioural and Brain Sciences* **13**, 471-518.

Hertz, J., Krogh, A. and Palmer, R. G. (1991) *Introduction to the Theory of Neural Computation*, Addison-Wesley.

Holland, J. H. (1986) Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In Michalski, R. S.,

Carbonell, J. G. and Mitchell, T. M. (eds.) *Machine Learning: An Artificial Intelligence Approach Volume II*, Morgan Kaufmann.

Hornik, K. Stinchcombe, M. and White, H. (1989) Multilayer feedforward networks are universal approximators. *Neural Networks* **2** (5), 359-366.

Horrocks, I., Patel-Schneider, P. F. and van Harmelen, F. (2003) From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics* **1** (1), 7-26.

Hu, W. and Qu, Y. (2008) Falcon-AO: A practical ontology matching system. *Web Semantics: Science, Services and Agents on the World Wide Web* **6** (3), 237-239.

Hu, W., Qu, Y. and Cheng, G. (2008) Matching large ontologies: A divide-and-conquer approach. *Data & Knowledge Engineering* **67**, 140-160.

Huhn, U. and Schulz, S. (2004) Building a very large ontology from medical thesauri. In Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. Berlin, Germany: Springer-Verlag, pp. 133-150.

Jean-Mary, Y. R., Shironoshita, E. P. and Kabuka, M. R. (2009) Ontology matching with semantic verification. *Web Semantics: Science, Services and Agents on the World Wide Web* **7** (3), 235-251.

Jones, D. M., Bench-Capon, T. J. M. and Visser, P. R. S. (1998) Methodologies for ontology development. In Cuena, J. (ed.) *IT & knows: Information Technologies and Knowledge Systems. Proceedings of a Conference Held as Part of the XV IFIP World Computer Congress. 31 August-4 September 1998, Vienna, Austria and Budapest, Hungary*. pp. 62-75. http://cgi.csc.liv.ac.uk/~tbc/publications/itknows.pdf <Accessed May 2017>

Kalfoglou, Y. and Schorlemmer, M. (2003) Ontology mapping: the state of the art. *The Knowledge Engineering Review* **18** (1), 1-31.

Klein, H. K. and Hirschheim, R. A. (1987) A comparative framework of data modelling paradigms and approaches. *The Computer Journal* **30** (1), 8-15.

Lippmann, R. P. (1987) An introduction to computing with neural nets. *IEEE ASSP Magazine, April 1987*, pp. 4-22.

Livet, P., Muller, J.-P., Phan, D. and Sanders, L. (2010) Ontology, a mediator for agent-based modeling in social science. *Journal of Artificial Societies and Social Simulation* **13** (1), 3. http://jasss.soc.surrey.ac.uk/13/1/3.html <Accessed May 2017>

Lovelace, R., Birkin, M., Ballas, D. and van Leeuwen, E. (2015) Evaluating the performance of iterative proportional fitting for spatial microsimulation: new tests for an established technique. *Journal of Artificial Societies and Social*

Simulation **18** (2), 21. http://jasss.soc.surrey.ac.uk/18/2/21.html <Accessed May 2017>

Moss, S. (2002) Agent based modelling for integrated assessment. *Integrated Assessment* **3** (1), 63-77.

Moss, S. and Edmonds, B. (2005) Sociology and simulation: statistical and qualitative cross-validation. *American Journal of Sociology* **110** (4), 1095-1131.

Moss, S. (2007) Alternative approaches to the empirical validation of agent-based models. *Journal of Artificial Societies and Social Simulation* **11** (1), 5. http://jasss.soc.surrey.ac.uk/11/1/5.html <Accessed May 2017>

Müller, J. P. (2010) A framework for integrated modeling using a knowledge-driven approach. In Swayne, D. A., Yang, W., Voinov, A. A., Rizzoli, A. and Filatova, T. (eds.) *Fifth Biennial International Congress on Environmental Modelling and Software, Ottawa, Canada*. http://www.iemss.org/iemss2010/papers/S21/S.21.08.A%20framework%20for%20integrated%20modeling%20using%20a%20knowledgedriven%20approach%20-%20JEAN-PIERRE%20MULLER.pdf <Accessed May 2017>

Ngo, D. and Bellahsene, Z. (2012) YAM++: A multi-strategy based approach for ontology matching task. In ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N. and Hernandez, N. (eds.) *Knowledge Engineering and Knowledge Management. 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings. Lecture Notes in Computer Science* **7603**, 421-425.

Object Modelling Group (2014) Ontology Definition Metamodel Version 1.1. *OMG Document Number: formal/2014-09-02* http://www.omg.org/spec/ODM/1.1/PDF/ <Accessed May 2017>

Oreskes, N. Shrader-Frechette, K. and Belitz, K. (1994) Verification, validation, and confirmation of numerical models in the earth sciences. *Science* **263** (5147), 641-646.

Perez, P., Dray, A., Dietze, P., Moore, D., Jenkinson, R., Siokou, C., Green, R., Hudson, S. L., Maher, L. and Bammer, G. (2009) An ontology-based simulation model exploring the social contexts of psychostimulant use among young Australians. *International Society for the Study of Drug Policy*. http://ro.uow.edu.au/smartpapers/36 <Accessed May 2017>

Polhill, J. G. (2015) Extracting OWL ontologies from agent-based models: A Netlogo extension. *Journal of Artificial Societies and Social Simulation* **18** (2), 15. http://jasss.soc.surrey.ac.uk/18/2/15.html <Accessed May 2017>

Polhill, J. G. and Gotts, N. M. (2009) Ontologies for transparent integrated human-natural systems modelling. *Landscape Ecology* **24**, 1255-1267.

Polhill, J. G., Sutherland, L.-A. and Gotts, N. M. (2010) Using qualitative evidence to enhance an agent-based modelling system for studying land use change. *Journal of Artificial Societies and Social Simulation* **13** (2), 10. http://jasss.soc.surrey.ac.uk/13/2/10.html <Accessed May 2017>

Radax, W. and Rengs, B. (2010) Prospects and pitfalls of statistical testing: insights from replicating the demographic prisoner's dilemma. *Journal of Artificial Societies and Social Simulation* **13** (4), 1. http://jasss.soc.surrey.ac.uk/13/4/1.html <Accessed May 2017>

Rossiter, S., Noble, J. and Bell, K. R. W. (2010) Social simulations: improving interdisciplinary understanding of scientific positioning and validity. *Journal of Artificial Societies and Social Simulation* **13** (1), 10. http://jasss.soc.surrey.ac.uk/13/1/10.html <Accessed May 2017>

Rumbaugh, J. (2003) Object-Oriented Analysis and Design (OOAD). In Ralston, A., Reilly, E. D. and Hemmendinger, D. (eds.) *Encyclopedia of Computer Science (4th Edition)*. Chichester, UK: John Wiley and Sons Ltd., pp. 1275-1279.

Schulze, J., Müller, B., Groeneveld, J. and Grimm, V. (2017) Agent-based modelling of social-ecological systems: Achievements, challenges, and a way forward. *Journal of Artificial Societies and Social Simulation* **20** (2), 8. http://jasss.soc.surrey.ac.uk/20/2/8.html <Accessed May 2017>

Shalizi, C. R. (2006) Methods and techniques of complex systems science: An overview. In Deisboeck, T. S. and Kresh, J. Y. (eds.) *Complex Systems Science in Biomedicine.* New York, NY, USA: Springer, pp. 33-114.

Shearer, R., Motik, B. and Horrocks, I. (2008) HermiT: A highly-efficient OWL reasoner. *OWLED 2008. OWL: Experiences and Directions. Fifth International Workshop, Karlsruhe, Germany, October 26-27, 2008*. http://webont.org/owled/2008/papers/owled2008eu_submission_12.pdf <Accessed May 2017>

Shvaiko, P. and Euzenat, J. (2013) Ontology matching: state of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering* **25** (1), 158-176.

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A. and Katz, Y. (2007) Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web* **5** (2), 51-53.

Sowa, J. (1999) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA, USA: Brooks/Cole.

Sure, Y., Staab, S. and Studer, R. (2004) On-To-Knowledge methodology (OTKM). In Staab, S. and Studer, R. (eds.) *Handbook on Ontologies*. Berlin, Germany: Springer-Verlag, pp. 117-132.

ten Broeke, G., van Voorn, G. and Ligtenberg, A. (2016) Which sensitivity analysis method should I use for my agent-based model? *Journal of Artificial Societies and Social Simulation* **19** (1), 5. http://jasss.soc.surrey.ac.uk/19/1/5.html <Accessed May 2017>

Thiele, J. C., Kurth, W. and Grimm, V. (2012) Agent-based modelling: Tools for linking NetLogo and R. *Journal of Artificial Societies and Social Simulation* **15** (3), 8. http://jasss.soc.surrey.ac.uk/15/3/8.html <Accessed May 2017>

Thompson, N. S. and Derr, P. (2009) Contra Epstein, good explanations predict. *Journal of Artificial Societies and Social Simulation* **12** (1), 9. http://jasss.soc.surrey.ac.uk/12/1/9.html <Accessed May 2017>

Troitzsch, K. G. (2009) Not all explanations predict satisfactorily, and not all good predictions explain. *Journal of Artificial Societies and Social Simulation* **12** (1), 10. http://jasss.soc.surrey.ac.uk/12/1/10.html <Accessed May 2017>

Troitzsch, K. G. (2015) What one can learn from extracting OWL ontologies from a NetLogo model that was not designed for such an exercise. *Journal of Artificial Societies and Social Simulation* **18** (2), 14. http://jasss.soc.surrey.ac.uk/18/2/14.html <Accessed May 2017>

Tsarkov, D. and Horrocks, I. (2006) FaCT++ description logic reasoner: system description. In Furbach, U. and Shankar, N. (eds.) *Automated Reasoning. Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006. Proceedings. Lecture Notes in Computer Science* **4130**, pp. 292-297.

Vapnik, V. N. and Chervonenkis, A. Y. (1971) On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications* **16**, 264-280.

Watkin, T. L. H., Rau, A. and Biehl, M. (1993) The statistical mechanics of learning a rule. *Reviews of Modern Physics* **65** (2), 499-555.

Windrum, P., Fagiolo, G. and Moneta, A. (2007) Empirical validation of agent-based models: alternatives and prospects. *Journal of Artificial Societies and Social Simulation* **10** (2), 8. http://jasss.soc.surrey.ac.uk/10/2/8.html <Accessed May 2017>

Winograd, T. (1972) *Understanding Natural Language*. Edinburgh University Press.

Wilensky, U. (1999) NetLogo. *Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL*. http://ccl.northwestern.edu/netlogo <Accessed May 2017>

Wood, S. N. and Augustin, N. H. (2002) GAMs with integrated model selection using penalized regression splines and applications to environmental modelling. *Ecological Modelling* **157** (2-3), 157-177.

Yang, G. and Feng, J. (2012) Database semantic interoperability based on information flow theory and formal concept analysis. *International Journal of Information Technology and Computer Science* **4** (7), 33-42.