Revilla E (2020). Individual and agent-based models in population ecology and conservation biology. Pp: 237-260 In: Population Ecology in Practice. First Edition. Eds Murray DL, Sandercock BK. Wiley-Blackwell, Hoboken, New Jersey. 472pp
ISBN/ISSN: 0470674148, 9780470674147

11

INDIVIDUAL AND AGENT-BASED MODELS IN POPULATION ECOLOGY AND CONSERVATION BIOLOGY

Eloy Revilla

Department of Conservation Biology, Estación Biológica de Doñana CSIC; Calle Américo Vespucio s/n; E-41092 Sevilla, Spain

Summary

Individual-based or agent-based models are a type of stochastic simulation models in which explicit agents or individuals interact with each other and the environment to generate system dynamics. The use of these models is linked to questions dealing with complex systems and is more akin to a research program than a method in itself, burrowing techniques from many different disciplines. First, the general aim and the questions to be addressed with the model, including the a priori expectations, must be explicit. The second step includes building the conceptual model based on the aim and the empirical and theoretical knowledge available. The conceptual model is then implemented in a core model which should be able to perform a single simulation run. The core model includes the definition of individuals and their traits, the functional relationships, the environment and its properties, the temporal and spatial domains, resolutions and boundary conditions and model scheduling. A single model run should produce an output that allows for an early evaluation of model consistency and that can be analyzed later on. At this stage, the conceptual model and the core model should be carefully documented. Finally, analyzing the model may require several steps, including model debugging at run time and an evaluation of the consistency of model behavior at the relevant

parameterizations and at extreme values; the evaluation of structural uncertainty and sensitivity analyses, including uncertainty analyses; the use of model selection techniques, if there are alternative model specifications; model validation and calibration, which consists of estimating model parameters by systematically comparing empirical and simulated data. Ultimately, the successful use of these models is highly dependent on having a clear aim and a good conceptual model. Given the complexity of the questions these models can address and the large flexibility that is allowed in analyzing them, this chapter is just a brief introduction to their construction and use.

11.1 Individual and agent-based models

Individual-based models (IBMs) belong to a broad class of stochastic simulation models in which the individuals (or more generally agents) of a population are explicit and identifiable, interacting under a set of rules within a given environment (DeAngelis and Mooij, 2005; Grimm and Railsback, 2005). Each individual is characterized by specific properties and state variables such as sex, age, reproductive status, body condition, the coordinates defining its spatial location or its genetic make-up. IBMs may range from very simple to extremely complex implementations. Nevertheless, the conceptual simplicity is one of the reasons why IBMs are becoming so pervasive in disciplines dealing with complex systems, such as astrophysics, cell biology, the social sciences or ecology (Gilbert, 2008; Grimm *et al.,* 2005). Complex systems are characterized by emergent properties generated by the interaction among its components and the environment. Typically, the behavior of those emergent properties is affected by stabilizing negative feedbacks and/or destabilizing positive feedbacks, as occurs with density dependent processes or with Allee effects. Conceptually, it is easy to grasp what IBMs are, as it is to build them if we have an intermediate command of a programming language. The difficult part is using these models in a way that is useful for our purposes and then communicating the methods and results to third parties in a clear and logical way. In this chapter I will try to help you in doing so.

Populations are just collections of *different* individuals. The uniqueness of individuals affects their realized fitness thus contributing in different amounts to the dynamics of the population to which they belong. Fortunately, the heterogeneity of individuals can be categorized into several main types that summarize the most relevant sources of heterogeneity in fitness, such us demographic classes, phenotypes or genotypes. In population ecology, we can take advantage of this structuring by averaging reproduction, survival and movement parameters within each of these groups and then describe or project population dynamics using those estimates. Nevertheless, class-specific demographic parameters vary through time and for individuals in different spatial locations, normally as a consequence of changes in relevant environmental variables.

Populations belong to the most challenging type of complex systems: adaptive systems, i.e. the responses of individuals can change (Grimm and Railsback, 2006). Apart from evolutionary responses, which may occur within a small number of generations making them relevant for population dynamics (DeAngelis and Mooij, 2005), individuals can show behavioral and other phenotypic responses (including memory, maternal effects or the effect of previous conditions within the domain of each individual), having the capacity to adapt their responses to environmental conditions in unexpected ways, making demographic functional responses very dynamic (Kuparinen and Merila, 2007; Doak and Morris, 2010). Methods dealing with complexity are especially useful for questions dealing with real populations. Nowadays, the major challenge of population ecology lies in having some forecasting capacity for populations composed of heterogeneous and adaptive individuals living in an environment which is also heterogeneous and dynamic in time and space.

11.1.1 What an individual-based model is and what it is not

The typical implementation of an IBM comes in the form of a computer program that executes, in a dynamic way, the processes describing the interactions among a set of individuals and their environment, generating relevant emergent properties at the population level, such as trajectories of population size in time, age, stage or sex distributions or distributions of density in space. Therefore, IBMs are simply a way to generate simulated data using stochastic numerical simulations. In itself it is not a method of analysis based on some statistical paradigm and therefore it departs from most of the methods described thus far in this book. To be of any use, the simulated data needs to be summarized by analyzing it in a similar fashion to that of field data, using everything we have learned so far, from how to generate and test sensible hypotheses, to estimating demographic parameters or analyzing time series and spatial structure. Therefore, the use of IBMs requires some a priori skills and an advanced research plan, including an adequate initial design for a clearly stated question, testing the general behavior of the model against empirical data and/or theoretical expectations and finally conducting some simulation experiments in which we systematically evaluate alternative scenarios in order to make some useful predictions.

Building an IBM requires software coding, either implicitly or explicitly. Nevertheless, coding is by no means the limiting factor when building an IBM. The main challenge is making explicit the question and designing a sensible and logical procedure to address it. Above all, using IBMs is an excellent way to make explicit our knowledge and assumptions in order to generate new hypotheses and predictions. It is therefore clear that IBMs are most relevant when aiming at complex questions for which other approaches are limited. To be able to do so we need a priori knowledge about how the system might work as well as information to be able to parameterize the model, even if using scenarios with hypothetical parameterizations (DeAngelis and Mooij, 2005; Grimm and Railsback, 2005).

11.1.2 When to use an individual-based model

The use of IBMs has increased significantly in the last few decades, and so has the diversity of research questions covered (Grimm, 1999). Models are often used to investigate complex questions, such as those having highly discordant spatio-temporal scales for different processes and patterns (generally local interactions generating data patterns at large scales), feedbacks and conditional parameter values affecting functional responses or strong impacts of spatial environmental heterogeneity on individual traits and responses. In many cases, the use of IBMs links population ecology to other disciplines, such as genetics, landscape ecology, behavioral ecology, ecotoxicology and economics. Typical studies range from population viability analysis of small populations for which demographic stochasticity is important, to management questions including the evaluation of different harvest regimes (Wiegand *et al.,* 1998, Whitman *et al.,* 2004), and questions dealing with population genetics, such as genetic structure or effective population size, and their relationship with demography and population viability (Storz *et al.,* 2002; Bruggeman *et al.,* 2010, Perez-Figueroa *et al.,* 2012). Authors often explore the role that different mechanisms can play at the population level under different environmental conditions, including physiological processes, such as individual energetics, growth and biomass dynamics or their interaction with diseases (Boyles and Willis, 2010; Buckley, 2008; Willis, 2007), as well as behavioral mechanisms, such as the link between individual behavioral responses and their impact on demographic parameters, the role of group living and sociality or spatial ecology and individual movements, including dispersal and how it impacts population dynamics (Goss-Custard *et al.,* 2006; Kramer-Schadt *et al.,* 2004; Rands *et al.,* 2006; Revilla *et al.,* 2004; Revilla and Wiegand, 2008; Stephens *et al.,* 2002; Tablado and Revilla, 2012). Finally, the use of IBMs in complex multi-specific questions, such as predation and community or disease dynamics is also relevant (Carlo and Morales, 2008; Ramsey and Efford, 2010; Rushton *et al.,* 2000; Schmitz, 2000; Wilkinson *et al.,* 2004).

11.1.3 Criticisms on the use of IBMs: Advantages or disadvantages

When first used, IBMs were heavily criticized along four main lines of thought. First, these models were described as too complex and therefore very data-hungry and prone to overfitting and error propagation problems. This critique has been based on a simplifying generalization and on some erroneous analyses (Beissinger and Westphal, 1998; Mooij and DeAngelis, 1999). If properly designed, calibrated and analyzed, IBMs are no more prone to those problems than any other applicable method (see Wiegand *et al.,* 2004b and the discussion and references therein). The generalization on over complexity is quite unfair since it is by definition not part of IBMs, but rather a consequence of addressing complex questions. Additionally, it confuses the definition of complexity used for statistical inference in statistics probability theory, defined by the number of parameters of a statistical model, with structural complexity under algorithmic theory. This leads to an axiomatic application of Occam's razor, which should be applied to empirically or theoretically supported process descriptions and when those descriptions are similarly supported by data. Only then should the model with fewer parameters be favored. The usefulness of a model is not given by the number of parameters, but rather its ability to address a question.

It is often assumed that the lower the number of parameters of a model the more generalizable the results, forgetting that the assumptions are also part of the model, and that to be able to make generalizations to other systems (not to say to make predictions) the set of assumptions must be sensible and comparable among systems. Structural realism is an important advantage of IBMs, especially in relation to model assumptions and even if model parameterization is not fully resolved or specified (Wiegand *et al.,* 2004b; Ajelli *et al.,* 2010). For example, the structural complexity of IBMs allows for the direct inclusion of demographic stochasticity with no need to parameterize it.

The remaining three criticisms are that IBMs are difficult to analyze, difficult to communicate, and, finally, the results are difficult to generalize in order to make inferences on the functioning of other systems (e.g., Bolker *et al.,* 2003). These points are relevant and represent the main challenge of using IBMs. The poor implementation of some early models, for some of which it appears as if the aim was to build the model itself, plus a poor documentation made the models too obscure and difficult to follow, not to mention replicate (Müller *et al.,* 2014). The only way to minimize those problems consists in using a research program aiming to understand how a complex system works (individual-based ecology *sensu* Grimm and Railsback, 2005). In doing so we should take advantage of the flexibility of IBMs, including the possibility of linking them to other methods, the capacity to make use of many sources of data with varying quality, including ancillary data, or the capacity to introduce difficult structures, such as covariation between model parameters, in a natural way. Finally, an important advantage of using IBMs is that if properly built, they force us to make explicit all the relevant knowledge on a population, including how different processes interact, and the capacity to generate predictions that are testable in the field.

11.2 Building the core model

11.2.1 Design phase: The question and the conceptual model

The first step in building an IBM is to identify and make explicit the general aim of the model. In the early days of IBMs it was not uncommon to find examples of models that were described with no further aim, consequently generating a lot of criticism. IBMs, as any other model, should be built to address a specific question. The general aim should be developed in the form of specific questions that can be directly linked to a priori predictions as well as data, both empirical and simulated. The theoretical and empirical context must be set, together with the general simplifying assumptions that are made a priori, such as no role for space or evolutionary processes.

The second step in the design phase consists in developing a conceptual model in which we summarize the knowledge in relation to the question to be addressed. At this stage it is quite useful to perform an in-depth review of the state of the art of the question, which should be made available to readers, either as part of the final manuscript or as a stand-alone publication. The conceptual model should make explicit the processes at the level of individuals that are known to affect some of their fitness traits (e.g. age mediated survival), the environmental factors modulating them (e.g. higher mortality at low temperatures) and the available parameter estimates including their central value, variability and uncertainty. It is also important, especially if we are dealing with a question related to a specific species and population, that we clearly differentiate the information coming from 1) general theory (including empirically derived heuristic patterns), 2) from species with a similar life-history and ecology, 3) from the same species in other populations and 4) from the focal population itself. This distinction will help us later on when defining model uncertainty, parameterizations and the alternative scenarios.

From the design phase we should have a shopping list with the working plan and the required pieces, including: 1) the individual traits (both those directly and indirectly linked with fitness); 2) processes and their parameters directly modifying individual traits (including rules and equations); 3) environmental processes and their parameters (indirectly affecting individual traits, their rules and equations); 4) a well-planned scheduling, i.e., how all those processes occur and integrate along the iterations of the model, that is, along the individuals in the population or through time; and, finally, 5) the emergent properties directly linked to the questions at hand (Fig. 11.1).

The design phase is critical and our final success will depend on doing a good job at this stage (Fig 11.1). It is also the most difficult part of the entire process, requiring some experience to master. The good news is that there is no single correct way to do it, and that we have a lot of freedom to follow our own preferences and style. Inexperienced researchers should consult relevant papers using IBMs (see section 11.1.2) and see how different authors deal with stating and breaking the general aim into questions and predictions and how they explain and justify their conceptual model. Building an IBM is about creating an explicit and dynamic representation of the available knowledge (conceptual model) on the relevant processes and their parameters affecting some variables of interest (emergent properties linked to the questions and predictions). We will have a chance to eventually be successful only if we have a clear question and a good conceptual model.

11.2.2 Implementation of the core model

The next step is the implementation of the conceptual model in a core model that by iteration of the processes generates some type of dynamics in a single simulation run. Normally, the core model is implemented using a programming language. The best language is the one you already know (or the one mastered by someone who can provide some support). There are so many potential choices that here we can only offer a brief field-guide to help you in deciding (Box 11.1), and make general recommendations that are useful across platforms and languages. There is no single best approach since different systems and languages have both advantages and disadvantages. Running simulations will require a modeling environment that allows for an efficient characterization of individuals and the proper integration across scales. Additionally, it is convenient that the system allows for debugging while coding and while running simulations, which will help in detecting errors and in the evaluation of model consistency (Fig. 11.1). Finally, the selected system should allow for fast simulation runs in order to be time-efficient in the analyses (Box 11.1).

11.2.3 Individuals and their traits

The population is a collection of individuals, but before creating any individual, we have to define their attributes, i.e., describe the traits and properties characterizing them, as defined in our conceptual model. For example, if we need to distinguish their sex, age and reproductive status, we will need to define those three identifiers. Even if two individuals have the same values for all traits, they must be unique and it should be possible to distinguish and find them within the population. Individual traits can be constant throughout their lifetime, for example their genetic makeup, or –depending on the taxa– their sex; or dynamic, if they change during the life of the individual, such as age or reproductive status (Box 11.2). The questions to be addressed with the model will help us in defining the initial population, which needs to be created from scratch at the beginning of a simulation. This population will have a given number of individuals, each with its own traits. As such, we create a population with a specific distribution of, for example, sexes, ages and statuses. Obviously the initial condition imposed by this population will have a profound impact on model dynamics: the dynamics generated by an initial population of 10 or 500 individuals will be quite different. Therefore the design and justification of the initial conditions should be thought out carefully and its impact analyzed.

11.2.4 Functional relationships

Individuals should interact in such a way that their fitness traits are affected. In classical population ecology we broadly distinguish between processes dealing with survival, reproduction and movement. Conceptualizing survival and reproduction as processes removing or adding individuals from the population is straightforward (Box 11.3). Movement is more complicated as it is a process mediating the addition or removal of individuals by migration. We can distinguish three types of processes directly affecting individuals: 1) those adding or 2) removing individuals and 3) those modifying individual traits, including responses to environmental conditions, behavioral responses and automatic modifiers of traits, such as aging (changing the age through time). They may range from very simple rules, for example if the maximum age is reached the individual must die deterministically, to complex sets of conditional equations such as a function calculating the probability of breeding as a function of local density and a set of environmental variables only if age and body condition allow for it. The possibilities are incredibly broad, but fortunately, we have a conceptual model at hand to identify what processes are potentially relevant.

Implementing functional relationships is normally done by programming subroutines, which is nothing more than a packed sequence of instructions that is executed whenever we call for it. Subroutines take different names in different languages (e.g., functions, procedures, methods) but they work in a similar way. Functional relationships are implemented by modifying variables (Box 11.4) with mathematical, logical and other types of operators as well as functions (for example, to obtain the absolute value of a floating number or to truncate it). In the case of complex equations we can make use of pre-coded libraries (which are subroutines in themselves) that can simplify the task. A key characteristic of subroutines in IBMs is that many of them need to go through the population, individual by individual, in order to perform the required calculations. For example, in order to apply an annual mortality rate we need to go through all individuals, one by one, and stochastically check if they can survive to the following year (Box 11.3).

11.2.5 The environment and its relevant properties

The environment represents the set of variables that act as direct or indirect modifiers of the traits of individuals. For example, if the probability of reproduction of a female depends on its age, the actual density and the amount of rain in that year with some specific parameters estimated with field data. Age is an individual property with its own dynamics whereas density and rain are external variables (for the focal individual). In this case, we need to calculate and keep track of population size and then calculate density during each simulation (Box 11.4). Note here that density dependence is probably one of the simplest impacts that the environment may have on the traits of each focal individual. The same applies for rain, which, depending on our needs, may be a predefined set of values (for example, a constant included in a one dimensional array of integer values, indexed from the first to the last year of data) or have its own dynamics depending on additional functions. Fixed environmental properties are included in the model as variables (with or without associated variability; Box 11.4); whereas in the case of dynamic environmental properties we need to include the processes describing the dynamics (rules, functional relationships and their parameters) in specific subroutines as we do with other processes. Environmental properties, which are also part of the initial condition, will have to be set up when starting the simulation.

11.2.6 Time and space: domains, resolutions, boundary conditions and scheduling

A critical element is how time and space are dealt with. Both are defined in all conceptual models, either implicitly or explicitly. In explicit definitions we need to keep track of them, either in continuous or discrete ways. If time and/or space are not explicit we still need to acknowledge them by clarifying the assumptions made on their reference domains. A domain is just the range of allowed values. Even in non-spatial models we have a spatial domain in the form of an assumption. Therefore, the first step is defining the temporal and spatial domains. Time is explicit in most cases (but not all), whereas both spatially implicit and explicit IBMs are common. For example, if we define the temporal domain of our model as 10 years (e.g. for a short-term reintroduction evaluation), we know that a simulation can run at most for that amount of time; or, if the spatial domain is 100 x 500 km that is the area in which our population occurs.

Within its domain, time can be represented by one or more temporal resolutions as required by the processes affecting individuals and the environment. The study of the interaction between processes at highly discordant temporal resolutions is essential for understanding the dynamics of complex systems (Grimm *et al.,* 2005). In the above example the 10 years can run in steps of one day or one year depending on the relevant processes. For example, in the case of univoltine species, reproduction can occur only once a year and therefore reproduction would require steps of one year. On the other hand, if we need to evaluate the role of the mortality imposed by short-term cold spells, we may think of a finer temporal resolution. Time is normally introduced as a conditional loop in which there is a counter that keeps track of the current time step (see subroutine for population dynamics in Box 11.3). If we have several temporal resolutions we can nest several conditional loops in a way that allows accounting for time as a clock does. For example, if we need days for survival and years for reproduction, we will code two nested loops, one counting years and another, within the previous one, counting days. Once the day loop runs for 365 days we start it again and the yearly loop moves to the next year.

In spatially explicit models we can proceed from simple to very complex descriptions of space (Box 11.5). Typically, we need explicit space when movement is a relevant process and therefore it needs to be implemented in subroutines, with rules and/or equations describing when, how and where individuals move. This is done by, changing the values of the traits describing the coordinates of individual location. Those subroutines tend to have fine-scale temporal resolutions to allow for individual movement decisions. All the rules and equations should be clearly specified (and justified) in the conceptual model (Nathan *et al.,* 2008). Associated with individual movement decisions is the concept of boundary conditions. What happens if individuals move to the edge of the spatial domain? Individuals can basically do two things, either be reflected back into the domain (as would be the case in a closed population moving within a fenced area, an island or an oversized spatial domain), or emigrate (i.e., leave the domain). If we implement emigration we may need to implement immigration as well. In some cases it is sufficient using a balanced emigration-immigration function by moving individuals back into the domain at the other end of the dimension they left (in a torus-like fashion). In any case, the best answer depends entirely on the system and the question at hand.

Finally, a critical concept we must think about carefully is that of scheduling, or how processes having different resolutions are nested and, for those with the same resolution, how they are ordered. Even in simple models, sometimes it is not easy answering questions such as what or who should be first, as is the case for survival and reproduction, in a model with only one temporal resolution (see for example the model in Box 11.3 and think about the effect of calling survival first instead of reproduction). In models with an implicit time, as occurs with some very short-term IBMs dealing with individual decision-making, or within a temporal resolution, we still need to define the order of interaction between individuals, that is, their cueing or implicit timing of inter-individual interactions. Different schedules affect model behavior and results. Again, the conceptual model is critical here as well as the explicit listing of how many temporal and spatial resolutions we have for each of the processes involved (Berec, 2002). Once you have a schedule it also helps plotting a diagram describing it (Figure 2).

11.2.7 Single model run, data input, model output

The core model can be used to run single simulations. As such, it is not of much use apart from demonstration or educational purposes in regard to our conceptual model. Most compilers allow for a process called debugging, which permits detecting the existence of programming errors, often locating the place where the code is flawed. Therefore, this debugging compilation will probably be the first manner of execution that we face, in the beginning, to our despair, but very much needed to obtain a clean and consistent core model. Nevertheless, debugging does not solve the inconsistencies that we introduced in the conceptual model or in the questions (Fig. 1).

In order to run the model we need to parameterize it by introducing values to all model parameters (Box 11.6) and defining the initial condition (initial population size and structure and the environmental setting). After running a simulation (or many) we need to obtain some output describing model behavior and predictions (Box 11.6). Remember that in the conceptual model we had identified simulated data directly linked to specific questions and their a priori predictions. IBMs are stochastic models and, therefore, the output variables will yield different results in different simulation runs with the exact same parameterization and initial condition. In order to estimate the probability distributions of each of the output data a number of simulation runs must be repeated with each parameterization. A reasonable rule of the thumb is enough runs to obtain stabilized estimates of the mean and standard deviation of the output variables.

11.3 Protocols for model documentation

At this stage we have a general aim that breaks into a set of specific questions and their potential responses based on a priori expectations, a conceptual model describing the system and the potentially relevant processes involved (and their parameters), and a description of how those processes drive the interactions between individuals, between those and the environment and the environmental dynamics itself, generating the dynamics of the population. We have implemented the conceptual model into a simulation model in what I have called the *core dynamic model*. At this stage, it is crucial to document what we did so far before the model gets too complex. During the process of building the model we probably needed to modify some parts and details of the conceptual model to accommodate the explicit way we built it and why we did so (Fig. 1). Once we start analyzing the model, we will probably need to revise both the conceptual and the core dynamic models again. A process of continuous refinement is normal and it is not a problem in itself. Nevertheless, and as complexity grows, we have to document what we have, even if it needs be modified later on.

Traditionally model documentation has ranged from simple verbal descriptions to very detailed descriptions and justifications, including pseudocode or even the full code of the model. Model documentation should run together with model building as it forces us to go through a process of thinking about how we are designing things and how all the components integrate. This documentation should include both model justification and a detailed description of its processes. For that reason, the refined version of the conceptual model, after the revision when constructing the model, should be the main part of the documentation.

Some general guidelines can help with properly informing about our work. We need to be as clear as possible about the general aim and the specific questions to be addressed, including the a priori predictions and the list of model behaviors and the variables dynamically predicted by the model that will be used in the analyses. If using field or theoretical data to compare with the predictions of your model, be as clear as possible about the methods used and the quality of those data sources. Make explicit all rules, equations and schedules included in each of the processes, with the help of graphs and other schemes if needed (Fig. 2). Use mathematical notation to declare equations and also rules (such as conditional probability or Boolean algebra notation). List model parameters, including constants, in association with the submodels they are implicated in, their description and the available estimates (this includes both the variability and the associated uncertainty), explaining and justifying the field and statistical methods used and/or the data sources. Make explicit all scales, domains, resolutions and how they integrate in each of the processes. Explain carefully how stochasticity is dealt with, including parameter sampling, randomization and any other decision that may affect the interpretation of the results (including for example data rounding and truncation). Finally, consider seriously publishing some version of your code, either in the form of annotated pseudocode (Box 11.3), the code of your core model, or all code produced for both the core model and the analyses (separated versions help in understanding what we did).

There have been several attempts to make explicit a list of minimum requirements to document IBMs in the form of model documentation protocols (Mooij and Boersma, 1996). The most popular is the Overview, Design concepts and Details (ODD) protocol presented by Grimm *et al.,* (2006), which has been updated and expanded by Grimm *et al.,* (2010) and by Topping *et al.,* (2010) who created the ODdox version for C++ code annotation and documentation. The result is a set of documents providing a heavily annotated and hyperlinked version of the ODD protocol linking model description to the source code. The ODD protocol or any other alternative can be used as a guideline to cross-check that we considered and described properly all the components of a model. The ODD protocol is a good way to organize and present information, but other alternatives maybe be more consistent with the aims and level of complexity of your model (Müller *et al.,* 2014).

11.3.1. The Overview, Design concepts and Details (ODD) protocol

The ODD aims to offer a standard that provides an ordered sequence of information that allows readers to follow the logic and details of any IBM (Grimm *et al.,* 2006; 2010). It first starts with general information in the Overview section (Table 1), described by three elements: the purpose of the model, the state variables and scales and finally a short overview of the processes and the scheduling. The next section, the design concepts, describes the strategic design of the model. The current version includes a list of eleven elements, ranging from emergence and adaptation to collectives or stochasticity. The list of elements is a bit arbitrary and it is not in a particularly relevant order. Go through them and build an ad hoc list by selecting the ones relevant for you. The final section goes into an explanation of the model in detail, including the initialization, the input data and finally, a detailed description of all processes. All sections and subsections of the ODD are articulated as groups of questions (Table 1). The final result is a document in which relevant details of the model are described. Nevertheless, following the guidelines of the ODD does not ensure that the explanations make sense, especially if your conceptual model is not consistent and well thought out. In the process of building your conceptual model you can use the ODD questions to cross-check what you might be skipping.

Grimm *et al.,* (2010) assume that a single protocol can suit all potential model implementations and that the ODD protocol should be strictly followed. However, the question of whether a single protocol can be applied to a variety of implementations built to address very different questions remains unresolved. My view is a bit more unorthodox because depending on the aims, we can find alternative ways to efficiently communicate our work. For example, in my view the clarity of the documentation of a model improves by clearly separating what belongs to the description of the core model from the description of the analyses. This includes different model parameterizations and initial conditions that are typically associated with specific analyses (which are normally several). In doing so, it is easier to understand the different steps, especially if the parameterization and initial conditions differ between analyses. Additionally, separating those two parts simplifies the distinction between what we consider as supported knowledge and the part that we will investigate in detail both in relation to model structure and parameterization.

11.4 Model analysis and inference

Analyzing a model is about understanding its behavior and its emergent dynamic properties under different conditions. The analysis of complex models is not a simple task. At this stage, the ecologist will use all her/his knowledge on experimental design and on statistical analyses, including the methods explained in this book. There is no single best way to analyze an IBM, with different approaches ideally yielding similar conclusions. Nevertheless, I offer some general guidelines to simplify the challenge. It is often difficult to distinguish between the phases of model building and model analysis because during the analyses we may be forced to redefine once again the initial conceptual model and the code, in another iteration of the modelling cycle (Fig 1; Grimm and Railsback, 2005). Normally we will follow a step by step program of analysis. I distinguish between four main steps. First, we need to go through a process of model debugging and consistency checking, followed by an evaluation of the consistency of model structure and a sensitivity analysis. Next come the steps of model selection, validation and calibration. Last, you should try to answer the questions that motivated the model within the inference constraints imposed by the previous results (Fig. 4).

11.4.1 Model debugging and checking the consistency of model behavior

Before going into your questions of interest, you should perform a thorough evaluation of model performance to detect errors arising from model design or implementation and determine if the behavior of the model makes sense. In this a priori checking you will detect many small problematic details and bugs that once removed will improve model consistency, saving a lot of time later on. Note that while writing the code of your model you were already debugging it at compilation time: any error appearing during compilation should have been corrected already (Box 11.4). Now we search for errors during execution time. The model should be able to run simulations with no errors during a single simulation run using a standard parameterization (the mean value and variability for all parameter estimates).

The next step consists of forcing model behavior with different combinations of parameters set at extreme values (for example, very low or high survival rates). Testing boundary conditions will force working with many zeros and with large numbers (including many individuals), thus making errors to appear. It is a good idea to repeat this step by step, going through the different processes before making overall extreme parameterizations of the model. Tests may generate problems by making forbidden or undefined calculations, such as floating point divisions by zero and other exceptions that the code does not handle properly. Many of the errors will be associated with exception handling, which depending on the language and compiler will be easy to solve. The following most important sources of errors will be associated with logical failures in scheduling and the way we introduce stochasticity into parameter values.

Simultaneous to model debugging during execution, it is important to look for biologically implausible behaviors, especially when working at extreme parameterizations. Before concluding that an interesting or unexpected behavior is a new finding, we must consider the possibility that it is associated with something incorrect in model specification or coding. The dynamics of the model should be consistent with the general expectations of the conceptual model. It is a good idea to use graphical output to cross-check the relevant output in run time, as well as saving simulated data together with parameters and tracking other data not directly related with the model aims and emergent properties, such as realized reproductive and mortality rates. All this information will serve as a log file, helping to determine whether an unexpected model behavior is due to a problem with design or programing, or if it is a new emergent result. Be sure to update the documentation of the model to describe the changes made in the conceptual or core models.

11.4.2 Model structural uncertainty and sensitivity analyses

The next step in analyzing an IBM should deal with setting the context in which to interpret the results: what are the limits for the inference? This step has two complementary sides, one related to model structural consistency, as defined by the processes and how they are integrated, and the other to the parameterization of those processes (Fig. 4). Thinking in the structural uncertainty of a model consists of specifying alternative definitions of the processes that we have implemented, such as using additive or multiplicative processes or different functions such as power or exponential laws. It is important when we do not have a good empirical description or theoretical justification for the choices. For example, imagine that based on empirical data we implemented a function in which survival is affected by temperature, but there is no data on which function is best and how it needs to be integrated with other factors such as density. If the main reason to build your model is addressing questions regarding the impact of temperature variation on some relevant population traits, it will be a good idea to think of alternative ways to implement the processes, such as an additive or multiplicative interaction with density. The idea is to create two or more alternative model structures that will be subject to sensitivity analyses. Further analyses will be repeated for each of the alternatives and the results compared for consistency under a model selection framework. Sensitivity analyses will help to gain confidence on how the specification of the model may affect inference. Structural uncertainty should be evaluated for processes that have some level of uncertainty and for which we expect, a priori, a relevant role on model behavior (Fig. 4).

In sensitivity analyses, we quantify how changes in the values of model parameters affect the value of the key output variables. This is achieved by repeatedly running the model with different parameterizations and measuring how the relevant outcomes respond. Depending on the aim of the analyses, we can differentiate between two different types: sensitivity analysis *sensu stricto* and uncertainty analysis. In sensitivity analyses we define the range of values to be explored using biologically realistic values for each model parameter that we want to explore. For example, the boundary conditions might set the parameter hypervolume; using parameters between the minimum and maximum values reported in the literature. In this way, we can explore the potential behaviors of the system under plausible conditions. Conversely, in uncertainty analysis we sample only within the existing uncertainty around each of the parameter estimates to determine the variability of the response of the model in relation to the available information. Typically for some parameters we do not have accurate estimates from the literature or from empirical data, for instance, the probabilistic parameters used in stochastic rules, and this uncertainty needs to be taken into account to avoid over-interpreting the results.

Sensitivity analysis is generally considered a key component of the quality evaluation of any model, for understanding the model itself and providing the context in which the rest of the results will be interpreted. For example, if the model aims to evaluate a conceptual hypothesis then the actual parameterization is not so relevant, whereas model behavior in a range of plausible conditions is. On the other hand, uncertainty analysis is particularly useful in indicating which parameters are candidates for additional research to narrow the degree of uncertainty in model results, and is a key component of models built for making predictions based on empirically estimated parameters. Something that is often overlooked in sensitivity analyses is the possibility of including how parameters interact by including covariation in parameter values. A final recommendation is avoiding sensitivity analyses using the central estimate of parameter values and an arbitrary small amount of variation (typically 5 or 10%) up and down. The range of values to be used should be well-justified.

In sensitivity or uncertainty analyses two general approaches are used depending on whether all parameters are considered simultaneously or not. In *local* and *one-at-a-time analyses* we sample the range of values of just one parameter while keeping all the others constant at their central estimate and then measuring to what extent the output of the model is affected. One-at-a-time approaches perform poorly when dealing with complex models such as IBMs and should in general be avoided (Saltelli and Annoni, 2010; but see Beaudouin *et al.,*. 2008). In *global* or *multivariate* sensitivity analyses we explore all the parameters simultaneously, repeatedly sampling the n-dimensional parameter hypervolume.

The sensitivity analysis will require a substantial amount of coding only for this purpose. Therefore, making a specific version of the model for this is a good idea. By coding loops, one for each parameter and with as many steps as values needed for each of them, you can run a global analysis at once even if you have a lot of parameters to sample. There are several ways to sample the parameter hypervolume, from simply randomly choosing parameter values (very inefficient) to a complete factorial sampling design, which may be reasonable for a reduced number of parameters. These approaches become computationally challenging for relatively small models. With just 10 parameters with 5 values each running with 100 simulation replicates to estimate the variability of the output requires 10^7 simulation runs. In these cases, we can use a more efficient Latin hypercube sampling (Iman and Helton, 2006). Briefly, this technique is a stratified sampling method commonly used to reduce the number of simulation runs necessary for sampling the parameter hypervolume. Each parameter is sampled using an even sampling method and then randomly combined sets containing all parameters are used to run the model. For each parameter the range of possible values is divided into non-overlapping intervals of equal probability size (Box 11.4). One value from each interval is chosen at random and this process is repeated for each parameter until we obtain a parameterization set. The key is that for every parameter each interval must be sampled only once until all intervals of all parameters have been used once. Then the process starts again. If the model is complex, it may be necessary to use a refined version of the Latin hypercube sampling that reduces the dimensionality of the problem by carefully analyzing some relevant processes before going into a simplified global analysis.

In the end, we obtain a dataset including the parameter values used and one or more relevant model predictions directly related with the questions (such as overall population size, density, growth rates, extinction probability, mean time to extinction or sex ratio). All this information needs to be summarized in order to obtain a picture of the differential role of the parameters and their associated uncertainty. The most basic way to do this is simply by using a partial rank correlation analysis (Segovia-Juarez *et al.,* 2004). A more inclusive approach is to run generalized regressions between model predictions (the average of the replicates for each parameterization) as dependent variable and model parameters as independent predictors (McCarthy *et al.,*. 1995). The resulting equations approximate the functions that relate the parameters of the simulation model to predictions in a simple way, while the standardized coefficients of the regression can be used to describe the sensitivity of model predictions to each of the input parameters (Revilla *et al.,* 2004; Revilla and Wiegand 2008). The generalized version of this approach is referred to as Gaussian process analysis in which the behavior of the simulation model in regard to each of its predictions is approximated by a Gaussian statistical model in which the predictors are the parameters of the simulation model (Dancik *et al.,* 2010). Remember that you need to report effect sizes and confidence intervals to give readers an idea of the magnitude and relative importance of each parameter effect. *P* values do not make sense here since the input parameters are known to generate the output, while the unlimited power provided by large simulated sample sizes makes their interpretation irrelevant.

Finally, we need to warn you against using sensitivity (or elasticity) analyses to make strong inferences about the actual factors driving the dynamics of a real population. These analyses do not necessarily tell you much about which parameters should be managed in the field. It specifies what each of the parameters does and the strength of the effect, so avoid making any definitive conclusion on what might be going on unless you have some empirical indication that the parameters identified as important in the sensitivity analyses are the ones that need to be managed. For example, the fact that adult survival is the most sensitive (or elastic) parameter in your model does not guaranty that the population is declining due to low adult survival; it could be entirely due to a lack of recruitment.

11.4.3 Model selection, validation and calibration

A bit trickier is comparing the outcome or outcomes of the model against a specific dataset. The comparison is usually made for different reasons, such as model selection, model validation and model calibration (Fig. 4, Table 2). If we are dealing with uncertainty in model structure, we will have alternative process specifications which can be assessed in their capacity to reproduce the observed data. In the case of validation, we typically have estimates of model parameters with their variability and uncertainty, which are then validated by evaluating their capacity to replicate an empirical dataset or some empirically observed behaviors, setting a credibility standard for that model structure, parameterization and question (Fig. 4). Calibration is a kind of model parameterization in which we estimate parameters from observed field data on model predictions by filtering out the parameterizations that do not match the data, by Gaussian process approximation or any other likelihood approximation (Hartig *et al.,* 2011). It is important to note that we leave model parameterization for the analysis-inference and not for model building since this step is very important in understanding how the model behaves. This is due to the fact that very often parameterization is first about defining and then reducing the dimensionality of the model before making any strong inference such as management recommendations. Model parameterization by calibration (or inverse modelling) may use no a priori information on the actual parameters, or may use the available information as priors under a Bayesian calibration framework (Hartig *et al.,* 2011). In mechanistic modelling, we assume that we can use information about the processes and how they integrate from other populations, whereas the parameters are just different realizations that we may observe. In model calibration, we can simultaneously perform the parameterization and the uncertainty analysis.

This step requires the systematic comparison of empirical and simulated data in order to decide which of the tested parameterization sets or model structures reproduce the empirical data in a reasonable way by calculating the probability of reproducing the field data with a given model structure and parameterization. Typically, we run simulations until we obtain a distribution of the frequencies of the simulated observations that the model structure and parameterization can generate and from them calculate the probability of observing the field values. The comparison between the observed and simulated data can be straightforward, as the difference or the sum of squared distances between the observed values and those obtained from the simulated data, or more efficient if we make the comparison only once against the summary statistics of the simulated frequency distribution (mean and variance). Conceptually, we can generalize all the alternative approaches as a kind of point-wise likelihood approximation of the goodness of fit of our model to the data (Hartig *et al.,* 2011). As such, we need to calculate the likelihood of observing the empirical data for each model structure and parameterization. The final goal is finding the structure and parameterization that maximizes that likelihood, thus obtaining a parameterization of the model with field data on model predictions, obtaining an estimate of the uncertainty (for example, by knowing how many alternative parameterizations match our threshold of fit) or simply helping us to select the model structure that is best supported by the available data (Fig. 4). Hartig *et al.,* (2011) review the different methods under a useful likelihood-based inference conceptual framework. The methods range from those that explicitly approximate the likelihood, such as approximate Bayesian computation, simulated (synthetic) pseudo-likelihoods or indirect inference, to those that allow calibrating the model without explicitly approximating the likelihood, such as pattern-oriented modelling or informal likelihoods (Beumont, 2010; Hartig *et al.,* 2011). The beauty of these methods is that the structural realism in the definition of processes at the right scales allows for inverse parameter estimation (Hartig *et al.,* 2011; 2014; Wood, 2010).

One of the classic ways to calculate the likelihood of obtaining the observed data given a model structure and parameterization makes use of central limit theorem, which allows us to calculate the probability of obtaining an empirical measurement from the summary statistics of the distribution of model outcomes for a given parameterization, if the simulated distribution can be approximated with a normal distribution (a parametric likelihood approximation, following the notation of Hartig *et al.,* 2011). For each model prediction we calculate a match-score, for example, a Z score using the mean and the standard deviation of the simulated replicates (Revilla *et al.,* 2004); while by setting different threshold probabilities for acceptance we can simultaneously evaluate multiple model predictions using a multicriteria approach, such as Pareto optimality assessment (Reynolds and Ford, 1999). Alternatively, we can use a Bayesian framework to calculate the posterior distribution and proceed in a similar manner (Beaumont, 2010; Hartig *et al.,* 2014). If the simulated frequency distribution generated by the model does not conform to a normal distribution (this typically occurs when using highly aggregated data which may generate multimodal distributions), then we may instead use a kernel density estimator to obtain a non-parametric estimation of the probability density function of the simulated distribution and subsequently calculate the probability of observing the empirical data from it (Tian *et al.,* 2007). There are cases in which the variability in the observed data is high due to measurement error but the predictions of the model for the same type of data shows lower variability. In these cases it is advisable adding a tractable error term (parametric or non-parametric) on the side of the observed data to account for noise (Hartig *et al.,* 2011). If we are evaluating alternative model structures, and therefore, we cannot be sure of the origin of the mismatch between observed and simulated data (structure, parameterization or stochasticity), it is advisable to use simpler measures of mismatch, such as the sum of squared distances between the observed and simulated data (informal likelihoods; Hartig *et al.,* 2011) or some kind of ad hoc rejection filtering under the pattern-oriented approach (Grimm *et al.,* 2005).

Pattern-oriented modeling, also termed rejection or performance filtering (Grimm *et al.,* 2005; Webb *et al.,* 2010; Hartig *et al.,* 2011), can be applied to models of dynamical systems. It is probably the most liberal approach in regard to model selection, validation and calibration, because it can also be used when the data to be adjusted (both the empirical and/or the simulated data) have complex distributions such as multimodal or multidimensional, or when the quality of the empirical data is poor or simply unknown. The method consists of defining criteria that allow classifying whether model structures or parameterizations match the observed data within a given explicit threshold, instead of calculating the actual likelihood of obtaining the observed value or a close enough value. The criteria used to define the thresholds can be diverse or even ad hoc, and may include some of the indexes of adjustment discussed above (for example, a mean squared difference or a Z score threshold). Additionally, we can use the error of the field data estimates to define the criteria. It allows using multiple ancillary data which in isolation do not contain much information, but that in combination can provide a robust approximation to constrain model behavior within the limits of the available information (Wiegand *et al.,* 2004b).

Potentially, the number of variables that may be included in the empirical dataset to be directly used in the comparison with simulated output can be large. Often we aggregate the available information in some way to obtain a simplified set of data that can be compared with the simulated output. These variables are referred to in the literature as patterns, state variables, output variables or simply summary statistics (Hartig *et al.,* 2011). The difficulty lies in deciding which of the many alternatives are statistically sufficient given the purpose of the model. The statistics need to convey information on the relevant properties of model dynamics. A good recommendation is to choose variables that operate at different spatial or temporal scales and hierarchical levels, including variables describing stationary and non-stationary dynamics (Grimm *et al.,* 2005; Wiegand *et al.,* 2004b; Wood, 2010). Nevertheless, the question behind your model should be the key when you to decide which data is relevant, obviously, within the limits imposed by the available empirical information.

All the methods discussed above require searching the potential parameter space in order to find the model structure or parameterizations best supported by data using some kind of numerical approximation (Bolker, 2008). In models with a reduced dimensionality, we can use a Latin hypercube sampling strategy. In more complex models, say above 20 parameters, depending on the availability of computing power, the programming language and how efficiently the model was coded, we will need a more efficient sampling strategy, such as Markov chain Monte Carlo strategies, including the Metropolis-Hastings and the Gibbs sampling algorithms, which start with an initial parameterization obtained from the parameter space, from which we generate a new parameterization by randomly moving a small amount within the parameter space. Then the likelihood, or similar, of the two consecutive parameterizations is compared, retaining the best one from which a new parameterization is obtained. There are lots of variants aiming to increase the speed, for example by reducing the correlation between consecutive parameterizations, and to avoid getting stacked in local likelihood maxima by going downhill with some probability. Another alternative is using sequential Monte Carlo approaches in which, starting with a set of parameterizations obtained from the whole parameter space, we calculate the point-wise likelihood and then weight each of them, for example by their normalized importance weight, according to their estimates. From this initial set we obtain a new set of parameterizations with probabilities according to their weights and repeat the process until some convergence criteria is met, such as that all parameterizations within the set are within a given likelihood threshold. Finally, we can consider using a numerical optimization algorithm when dealing with multiple data to be fitted under a pattern-oriented approach (Table 2). Hartig *et al.,* (2011) provide pseudocode algorithms for some of these numerical sampling methods. Applying these methods is most efficiently done by programming the routines within the coding environment. The methods in themselves are not complicated (though the specific jargon is) but require extensive coding. Remember making a specific version of the model for the purpose of validation and calibration. A potentially less efficient alternative is generating the simulated datasets and then using some of the algorithm implementations available within R.

11.4.4 Answering your questions

At this stage, and after all the work done, we should have a clear idea of the questions to answer. The potential uses of IBMs are broad and flexible, as occurs with other stochastic simulation models, making difficult to summarize their uses (see examples given in *16.1.2*). The first and most basic use consists of reviewing and integrating the available knowledge on a system. This is basically done by building the conceptual model and its implementation in a core model plus the sensitivity analysis over the biologically plausible parameter space and a validation of the model with independent data. We must give all the available information, making clear what is supported by knowledge and data and what are the assumptions and hypotheses which should be investigated further. From this initial step, the following typical use of IBMs consists of gaining new knowledge on how a system usually works, often evaluating the predictions of theoretical models and empirical generalizations for population regulation, movement, density dependence or interspecific interactions such as predation or diseases. Last, practical applications represent a broad field of use, including population viability analyses, the evaluation of alternative management scenarios for conservation, population control or exploitation, the evaluation of strategies to control diseases or measuring the impact of infrastructures on interpopulation connectivities, just to mention a few.

All these uses have in common the description of model behavior under different scenarios. A scenario is defined by a model structure, an initial condition and a parameterization, which also includes the space definitions used in spatially explicit models, normally as maps. For the scenario we obtain frequency distributions of the relevant model outcomes by running multiple stochastic simulations. The simplest approach is just a qualitative or quantitative description of those outcomes, for example, by plotting the results in figures. It is much more common that we need to compare the results of one scenario against other scenarios, empirical data or theoretical expectations in a qualitative and/or quantitative way, as discussed in the previous section. Comparing the output of the model for alternative scenarios is more or less straightforward, especially if what we need is the relative evaluation against a desired standard. For example, we may need to evaluate alternative hunting strategies to estimate maximum yield, to reduce interannual variability in population size, or to minimize extinction risk. We can also use statistical descriptions to compare the distributions of outcomes for the different scenarios. The comparison of multiple scenarios, such as management alternatives, needs to be carefully thought out under the standard framework of experimental design (the virtual ecologist approach; Zurell *et al.,* 2010).

Finally, one important issue to consider when designing the experiments is the dependence of model behavior on both its current and past states (model hysteresis). The initial conditions or a perturbation often impose a transient state phase after which the system may reach a steady state with stationary stochastic dynamics, which occurs when the dynamic properties of the model do not change over time, with the frequency distributions of model outcomes remaining stable. Depending on the aims, we may need to focus on the non-stationary dynamics, for example, when studying the impact of an event or perturbation, such as the success rate of different reintroduction scenarios varying in the number of animals released (Kramer-Schadt *et al.,* 2005) or a PVA affected by the initial conditions imposed by an empirical estimate of population size and structure (Wiegand *et al.,* 1998). We can also focus on the steady state phase, as we do when calculating the intrinsic mean time to extinction in PVA (Grimm and Wissel, 2004), or on both, transient and steady phases, for example, when investigating the impact of different management activities starting with an observed initial population size (Wiegand *et al.,* 2004a).

11.5 Final thoughts

This chapter is a bit different from the others. More than discussing a specific method with a lot of examples, it deals with a research approach that can be implemented in many alternative ways to address a potentially very broad range of questions. As such, it borrows methods from many disciplines, including not only ecology, but also statistics, complex systems and algorithmic theories and software engineering. I did not intend to present a thorough review of the literature in regard to examples of IBM implementations and applications. Instead, I aimed to provide an overview of the whole process, from the beginning to the end of the research program, focusing on those parts that might be more challenging for newcomers and hopefully providing some useful guidelines. Using IBMs is by no means easy. The challenge remains in having a good conceptual model and very clear questions early on. Analyzing the model requires some experience in order not to be overwhelmed or lost in irrelevant detail. As with using any other approach that relies on programming, the learning curve may be steep, but it should lead somewhere, and knowing where to go is on the side of the user. Remember that, by itself, building a model is not the question to answer.

I provide some toy models in the online materials. They are built merely to illustrate one of the many different ways you may choose to start coding an IBM. This should help you to feel more comfortable with how IBMs are built. Those examples are not core models, just out-of-the-box toy models for you to play with, modify, corrupt, modify again and in this manner learn a bit more about the logic behind this research approach. Then, with the help of this chapter and the methods presented in the rest of the book, you should be able to address your research questions.

References

Ajelli M, Gonçalves B, Balcan D, *et al*. (2010) Comparing large-scale computational approaches to epidemic modeling: agent-based versus structured metapopulation models. *BMC Infectious Diseases* **10**(1), 190.

Beaudouin, R, Monod G, Ginot V (2008) Selecting parameters for calibration via sensitivity analysis: An individual-based model of mosquitofish population dynamics. *Ecological Modelling* **218**, 29-48.

Beaumont MA (2010) Approximate bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution and Systematics* **41**, 379–406.

Beissinger SR, Westphal MI (1998). On the use of demographic models of population viability in endangered species management. *Journal of Wildlife Management* **16**, 821–841.

Berec L (2002) Techniques of spatially explicit individual-based models: construction, simulation, and mean-field analysis. *Ecological Modelling* **150**, 55-81.

Bolker B (2008) *Ecological models and data in R*. Princeton University Press, Princeton.

Bolker B, Holyoak M, Křivan V, *et al*. (2003) Connecting theoretical and empirical studies of trait-mediated interactions. *Ecology* **84**, 1101-1114.

Boyles JG, Willis CKR (2010) Could localized warm areas inside cold caves reduce mortality of hibernating bats affected by white-nose syndrome? *Frontiers in Ecology and the Environment* **8**, 92-98.

Bruggeman D, Wiegand T, Fernandez N (2010) The relative effects of habitat loss and fragmentation on population genetic variation in the red-cockaded woodpecker (*Picoides borealis*). *Molecular Ecology* **19**: 3679-3691.

Buckley LB (2008) Linking traits to energetics and population dynamics to predict lizard ranges in changing environments. *The American Naturalist* **171**, E1-E19.

Carlo TA, Morales JM (2008) Inequalities in fruit-removal and seed dispersal: consequences of bird behaviour, neighbourhood density and landscape aggregation. *Journal of Ecology* **96**, 609–618.

Dancik GM, Jones DE, Dorman KS (2010) Parameter estimation and sensitivity analysis in an agent-based model of *Leishmania major* infection. *Journal of Theoretical Biology* **262**, 398–412.

DeAngelis DL, Mooij WM (2005) Individual-based modeling of ecological and evolutionary processes. *Annual Review of Ecology Evolution and Systematics* **36**, 147–68.

Doak DF, Morris WF (2010) Demographic compensation and tipping points in climate-induced range shifts. *Nature* **467**, 959-962.

Gilbert N (2008) *Agent-based models*. Sage, No. 153.

Goss-Custard J, Burton N, Clark N, *et al*. (2006) Test of a behavior-based individual-based model: Response of shorebird mortality to habitat loss. *Ecological Applications* **16**, 2215-2222.

Grignard A, Taillandier P, Gaudou B, *et al*. (2013) *GAMA* 1.6: Advancing the Art of Complex Agent-Based Modeling and Simulation. *The 16th International Conference on Principles and Practices in Multi-Agent Systems (PRIMA)* **8291**, 242-258.

Grimm V (1999) Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future? *Ecological Modelling* **115**, 129-148.

Grimm V, Railsback SF (2005) *Individual-based Modeling and Ecology*. Princeton Series in Theoretical and Computational Biology, Princeton.

Grimm V, Railsback SF (2006) Agent-Based Models in Ecology: Patterns and Alternative Theories of Adaptive Behaviour. In: *Agent-based Computational Modelling, Contributions to Economics*. Pp: 139-152.

Grimm V, Revilla E, Berger U, *et al*. (2005) Pattern-oriented Modeling of Agent-based Complex Systems: Lessons from Ecology. *Science* **310**, 987-991.

Grimm V, Berger U, DeAngelis DL, *et al*. (2010) The ODD protocol: A review and first update. *Ecological Modelling* **221**, 2760-2768.

Grimm V, Berger U, Bastiansen F, *et al*. (2006) A standard protocol for describing individual-based and agent-based models. *Ecological Modelling* **198**, 115-126.

Grimm V, Wissel C (2004) The intrinsic mean time to extinction: a unifying approach to analyzing persistence and viability of populations. *Oikos* **105**, 501-511.

Hartig F, Calabrese JM, Reineking B, *et al*. (2011) Statistical inference for stochastic simulation models – theory and application. *Ecology Letters* **14**, 816–827.

Hartig F, Dislich C, Wiegand T, *et al*. (2014) Technical Note: Approximate Bayesian parameterization of a process-based tropical forest model. *Biogeosciences* **11**, 1261-1272.

Iman RL, Helton JC (2006) An Investigation of Uncertainty and Sensitivity Analysis Techniques for Computer Models. *Risk Analysis* **8**, 71-90.

Kramer-Schadt S, Revilla E, Wiegand T (2005) Lynx reintroductions in fragmented landscapes of Germany: projects with a future or misunderstood wildlife conservation? *Biological Conservation* **125**, 169-182

Kramer-Schadt S, Revilla E, Wiegand T, *et al*. (2004) Fragmented landscapes, road mortality and patch connectivity: modelling influences on the dispersal of Eurasian lynx. *Journal of Applied Ecology* **41**, 711-723.

Kuparinen A, Merila J (2007) Detecting and managing fisheries-induced evolution. *Trends in Ecology and Evolution* **22**, 652–659.

Letcher BH, Priddy JA, Walters JR, *et al*. (1998) An individual-based, spatially-explicit simulation model of the population dynamics of the endangered red-cockaded woodpecker, *Picoides borealis*. *Biological Conservation* **86**, 1-14.

Liu J, Dunning Jr JB, Pulliam HR (1995) Potential Effects of a Forest Management Plan on Bachman's Sparrows (*Aimophila aestivalis*): Linking a Spatially Explicit Model with GIS. *Conservation Biology* **9**, 62-75.

Luke S, Cioffi-Revilla C, Panait L, *et al*. (2005) MASON: A Multi-Agent Simulation Environment. *Simulation: Transactions of the society for Modeling and Simulation International* **82**, 517-527.

Macal CM, North MJ (2009) Agent-based modelling and simulation. In: *Proceedings of the 2009 Winter Simulation Conference*. Eds.: M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. Pp: 86-98.

Macal CM, North MJ (2010) Tutorial on agent-based modelling and simulation. *Journal of Simulation* **4**, 151-162.

McCarthy MA, Burgman MA, Ferson S (1995) Sensitivity analyses for models of population viability. *Biological Conservation* **73**, 93–100.

Mooij WM, Boersma M (1996) An object-oriented simulation framework for individual-based simulations (OSIRIS): *Daphnia* population dynamics as an example. *Ecological Modelling* **93**, 139–53

Mooij WM, DeAngelis DL (1999) Error propagation in spatially explicit population models: a reassessment. *Conservation Biology* **13**, 930–933.

Müller B, Balbi S, Buchmann CM, *et al*. (2014) Standardised and transparent model descriptions for agent-based models – current status and ways ahead. *Environmental Modelling & Software* **55**, 156-163.

Nathan R, Getz WM, Revilla E, *et al*. (2008) A Movement Ecology Paradigm for Unifying Organismal Movement Research. *Proceedings of the National Academy of Sciences USA* **105**, 19052-19059.

Perez-Figueroa A, Wallen R, Antao T, *et al*. (2012) Conserving genomic variability in large mammals: Effect of population fluctuations and variance in male reproductive success on variability in Yellowstone bison. *Biological Conservation* **150**, 159-166.

Railsback SF, Lytinen SL, Jackson SK (2006) Agent-based simulation platforms: Review and development recommendations. *Simulation* **82**, 609-623.

Ramsey DSL, Efford MG (2010) Management of bovine tuberculosis in brushtail possums in New Zealand: predictions from a spatially explicit, individual-based model. *Journal of Applied Ecology* **47**, 911-919.

Rands SA, Pettifor RA, Rowcliffe JM, *et al*. (2006) Social foraging and dominance relationships: the effects of socially mediated interference. *Behavioral Ecology and Sociobiology* **60**, 572-581.

Revilla E, Wiegand T (2008). Individual movement behavior, matrix heterogeneity, and the dynamics of spatially structured populations. *Proceedings of the National Academy of Sciences USA* **105**, 19120-19125.

Revilla E, Wiegand T, Palomares F, *et al*. (2004) Effects of matrix heterogeneity on animal dispersal: From individual behavior to metapopulation-level parameters. *The American Naturalist* **164**, E130-E153.

Reynolds JH, Ford D (1999) Multi-criteria assessment of ecological process models. *Ecology* **80**, 538-553.

Rushton SP, Lurz PWW, Gurnell J, *et al*. (2000) Modelling the spatial dynamics of parapoxvirus disease in red and grey squirrels: a possible cause of the decline in the red squirrel in the UK? *Journal of Applied Ecology* **37**: 997-1012.

Saltelli A, Annoni P (2010) How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software* **25**, 1508-1517.

Schmitz OJ (2000) Combining field experiments and individual-based modeling to identify the dynamically relevant organizational scale for a field system. *Oikos* **89**, 471–484.

Segovia-Juarez JL, Ganguli S, Kirschner D (2004) Identifying control mechanisms of granuloma formation during *M. tuberculosis* infection using an agent-based model. *Journal of Theoretical Biology* **231**, 357–376.

Stephens PA, Frey-Roos F, Arnold W, *et al*. (2002) Model complexity and population predictions. The alpine marmot as a case study. *Journal of Animal Ecology* **71**, 343-361.

Storz J, Ramakrishnan U, Alberts S (2002) Genetic effective size of a wild primate population: Influence of current and historical demography. *Evolution* **56,** 817-829.

Tablado Z, Revilla E (2012) Contrasting effects of climate change on rabbit populations through reproduction. *PLoS One* **7**(11): e48988.

Tian T, Xu S, Gao J, *et al*. (2007) Simulated maximum likelihood method for estimating kinetic rates in gene expression. *Bioinformatics* **23**: 84–91.

Topping CJ, Hansen TS, Jensen TS, *et al*. (2003) ALMaSS, an agent-based model for animals in temperate European landscapes. *Ecological Modelling* **167**, 65-82.

Topping CJ, Hoye TT, Olesen CR (2010) Opening the black box: Development, testing and documentation of a mechanistically rich agent-based model. *Ecological Modelling* **221**, 245-255.

Webb CT, Hoeting JA, Ames GM, *et al*. (2010) A structured and dynamic framework to advance traits-based theory and prediction in ecology. *Ecology Letters* **13**, 267-283.

Whitman K, Starfield AM, Quadling HS, *et al*. (2004) Sustainable trophy hunting of African lions. *Nature* **428**, 175-178.

Wiegand T, Knauer F, Kaczensky P, *et al*. (2004a) Expansion of brown bears (*Ursus arctos*) into the eastern Alps: a spatially explicit population model. *Biodiversity and Conservation* **13**, 79-114.

Wiegand T, Revilla E, Knauer F (2004b). Dealing with uncertainty in spatially explicit population models. *Biodiversity and Conservation* **13**, 53-78.

Wiegand T, Naves J, Stephan T, *et al*. (1998) Assessing the risk of extinction for the brown bear (*Ursus arctos*) in the Cordillera Cantabrica, Spain. *Ecological Monographs* **68**, 539-570.

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

Wilkinson D, Smith CG, Delahay RJ, *et al*. (2004) A model of bovine tuberculosis in the badger Meles meles: an evaluation of different vaccination strategies. *Journal of Applied Ecology* **41**, 492-501.

Willis J (2007) Could whales have maintained a high abundance of krill? *Evolutionary Ecology Research* **9**, 651-662.

Wood SN (2010) Statistical inference for noisy nonlinear ecological dynamic systems. *Nature* **466**, 1102-1104.

Zurell D, Berger U, Cabral JS, *et al*. (2010) The virtual ecologist approach: simulating data and observers. *Oikos* **119**, 622-635.

Table 1. The Overview, Design concepts and Details (ODD) protocol (modified from Grimm *et al.,* 2006; 2010).

Elements	Questions
Overview	*Context and general information*
1. Purpose	What is the purpose of the model?
2. Entities, state variables and scales	What entities (e.g., individuals, collectives) are in the model? By what state variables (attributes and traits) are these entities characterized? What are the temporal and spatial resolutions and domains of the model?
3. Process overview and scheduling	Who (entity) does what, and in what order? When are state variables updated? How is time modeled, as discrete steps or as a continuum over which both continuous processes and discrete events can occur?
Design	*Strategic considerations*
4. Design concepts	
4.1. basic principles	Which theories, hypotheses, assumptions or modeling approaches are behind a model's design? How were they taken into account? Are they used in submodels or at the system level? Will the model provide insights into the basic principles themselves?
4.2. emergence	What model results are expected to vary in complex and perhaps unpredictable ways when particular characteristics of individuals or their environment change? Are there other results that are more tightly imposed by model rules and hence less dependent on interactions?
4.3. adaptation	What adaptive traits do the individuals have? What rules do they have for making decisions or changing behavior in response to changes in themselves or their environment? Do these traits explicitly seek to increase some measure of individual success regarding its objectives, or, instead, cause individuals to reproduce previously observed behaviors?
4.4. objectives	If adaptive traits explicitly act to increase some measure of individual fitness, what exactly is that objective and how is it measured? When individuals make decisions by ranking alternatives, what criteria do they use?
4.5. learning	Do individuals change their adaptive traits over time as a consequence of experience? If so, how?
4.6. prediction	How do individuals predict the future conditions (either environmental or internal) they will experience? What internal models do they use to estimate future conditions or the consequences of their decisions? What tacit or hidden predictions are implied in these internal model assumptions?
4.7. sensing	What internal and environmental state variables (including those of other individuals) are individuals assumed to sense and consider in their decisions? Are there mechanisms by which individuals obtain information, or are they assumed to know these variables?
4.8. interaction	What kinds of interactions among agents are assumed? Are there direct interactions in *which* individuals encounter and affect others, or are interactions indirect? If the interactions involve communication, how is it represented?
4.9. stochasticity	What processes are modeled as random or partly random? Is stochasticity used to reproduce variability in processes for which the actual causes of the variability are unknown or not relevant? Is it used to model events or behaviors with a specified probability?
4.10. collectives	Are there social networks? If so, is its structure imposed (a priori additional entity) or emergent? Are collectives affecting, or been affected by the individuals?
4.11. observation	What data are collected from the simulations for testing, understanding, and analyzing the model? How and when are they collected?
Details	*Detailed technical description*

5. Initialization	What is the initial state of the model at the beginning of a simulation run? Is initialization always the same, or is it allowed to vary among simulations? Are the initial values chosen arbitrarily or based on data?
6. Input data	Does the model use input from external sources such as data files or other models to represent processes that change over time?
7. Submodels	What, in detail, are the processes listed in point 3? How were they designed, parameterized and tested? What are their parameters, dimensions and reference values?

Table 2. Some issues to consider when comparing empirical and simulated data for model selection, validation and calibration.

Data	
Key data	Empirical data directly related with the questions to be answered with the model.
Ancillary or secondary data	Empirical data not directly related with the questions. It contains information useful in model selection and calibration. Often corresponds to data at discordant spatiotemporal scales.
Estimates	Key and secondary data can be quantitative, including point estimates and their uncertainty and variability, or qualitative, such as trends
Summary statistics	Aggregation of data into new simplified yet informative statistics (for example calculating a growth rate from a raw series of census data). This is often done to simplify the comparison between data and predictions.
Single vs multiple	The amount of data can vary from a single key variable to multiple key variables and secondary data.
Predictions	
Symmetry	We need to calculate as model output the same key and secondary predictions as with the empirical data.
Single parameterization	For a given parameterization we generate a frequency distribution of model predictions by repeating a number of simulations with the parameterization.
Output formats	Predictions can be obtained as graphical outputs to visualize the results and saved into files. It is convenient saving the parameterization within the output files.
Multiple parameterizations	Often we need to repeat the process for multiple parameterizations obtained by moving across the parameter space.
Comparisons	
The logic	Systematically compare data and predictions to estimate the likelihood of reproducing the observed data with a given parameterization and model structure.
Types of comparisons	Rejection filtering by using pattern oriented modelling or informal likelihoods
	Direct calculation of the likelihood by running a sufficiently large number of simulations
	Informal likelihoods (e.g. sum of squared differences between data and predictions)
	Non-parametric likelihood approximations (e.g. kernel density estimation)
	Parametric likelihood approximations (e.g. Z scores)
	Approximate Bayesian computation
Methods to define parameterizations	Systematic search of the parameter space when the number of parameters is low
	Latin hypercube sampling for more complex models
	Markov chain Monte Carlo strategies: Metropolis-Hastings and Gibbs sampling algorithms and their variants.
	Sequential Monte Carlo approaches, also known as particle filters or bootstrap filters
	Numerical optimization methods such as genetic algorithms, simulated annealing, simplex algorithm or support vector machine algorithms

Box 11.1 Programs and software: A field guide to some individual based coding environments

We can use three types of approaches: using software that allows for scripting using interpreted languages, general multipurpose programing languages that allow for object oriented programing, or specialized development environments created specifically to build agent based models.

Approaches useful to build demonstrator models

We can create IBMs using software which allows for scripting, as is possible in some spreadsheets such as Gnumeric, LibreOffice Calc or proprietary MsExcel, noting that you need some knowledge of Visual Basic for Applications, Python or any other supported scripting language to program the macros (e.g., Macal and North 2010Raisl). These implementations are useful as demonstrators for learning concepts and teaching or for implementing structurally very simple IBMs for which the analyses are simple. We can also build IBMs in environments that are very efficient in making generalized scalar operations such as in vectorial or array programming languages, such as R or Matlab, or even in more eclectic languages such as Wolfram (running in proprietary Mathematica).

R is a software platform that allows for the efficient manipulation and analysis of relatively small datasets. It is so flexible that we can also build IBMs with it. However, doing so is only reasonable for learning purposes or when dealing with very simple IBMs (few parameters and individuals). R uses array programing, operating with all the data simultaneously, making the processing of large datasets inefficient. Therefore, it is slow and resource hungry in dealing with the data we create when, for example, running a sensitivity analyses across many-dimensional spaces. It is also an interpreted language, i.e., does not compile the commands we write into machine code, making simulations much slower than other alternatives.

General purpose development environments

This group refers to compiling object-oriented programming languages that allow programming totally ad hoc models. Normally the source code is written within a computer program called *compiler* that transforms the source language into a machine compatible language that can be executed by the computer. This approach is more efficient than interpreted languages, allowing for much faster simulations. Creating individuals is straightforward using *objects* or *classes*. After compilation we can obtain a range of possibilities, from a self-contained executable file to a sophisticated application with a detailed Graphical User Interface (GUI, normally created by using *Forms*) that may allow for interaction with the user during the initialization (e.g., for parameterization), a graphical inspection of model behavior during run time and also the exploration of the results. We can cite *C++*, *Python* (to some extent) or *Java* as general languages, with different derivations of *Fortran* and *Object Pascal* being very popular in academic and scientific applications. All of them have many compilers available. If you have some experience programing this would probably be your best way to proceed.

To run the model we have several alternatives, very much dependent on the language we are using and the environment (compiler and operating system). The most basic is a batch-like mode in which, after asking for execution (e.g., by clicking in the exe file created by the compiler after a successful compilation), all the code is executed at once with no further intervention on our part. In most modern programing languages we interact with a compiler that includes prewritten components (library-like) that can be used and reused allowing for fast model construction and deployment. Forms are the most basic of such components when running the program. They create a window that allows for interaction between the user and the model at run time. Many other components can be used, including buttons to be inserted in the form which execute some code when we click on them. Forms and other components with which we interact are part of the GUI of our model. If, for example, the pseudocode in Box 11.3 was written in a compiler allowing for forms, we could add to it a button which on a click would run the subroutine for population dynamics.

Specialized development environments:

These are just implementations built using general programing languages but that offer through an Application Programming Interface (API) access to precoded libraries that can simplify the initial work of making explicit the conceptual model (Railsback *et al.,* 2006). Using a specific environment would save you a lot of time if you have no experience programing. Running the model in specialized development environments is straightforward, just follow the program instructions. Specific environments for building IBMs have their own detailed documentation and many examples to build upon. A non-exhaustive list would include:

ALMaSS, Animal, Landscape and Man Simulation System. A complex highly specific model, with detailed implementations built for different species (e.g., voles, skylarks). The model is spatially explicit, including individual movement behavior, a landscape model that can be dynamic and a weather simulator. Open source project written in C++. Topping *et al.* (2003). http://ccpforge.cse.rl.ac.uk/gf/project/almass/

GAMA. A highly flexible system that allows for the development of complex spatially explicit models of potentially very large populations. The conceptual model is coded in GAML language, which is a derivative of XML. Allows for calling R and SQL code using several DBMS. The user interface is based on the Eclipse platform (which is itself mostly written in Java). Grignard *et al.,* (2013). http://code.google.com/p/gama-platform/

Repast. A set of open source platforms to perform agent-based modelling and simulations, including spatially explicit models. Different implementations either including Java or C++ coding systems. Allows for fast simulations and large and very complex models to be built. Very complete and with many tools available. Macal and North (2009). http://repast.sourceforge.net/

Mason. Multiagent simulation of neighborhoods. It is a discrete event agent based simulation platform implemented in Java (requires experience with this language). It is fast, flexible and portable across machines, with good capacity to run in batch mode with no visualization. Luke *et al.,* (2005). http://cs.gmu.edu/~eclab/projects/mason/

NetLogo. A very intuitive and easy to use system to develop simple grid-based models. Recommended for people with no programing experience. Based on a language derived from Logo (but built in Java), with many primitives (built-in commands). Includes a collection library with many ecological model examples. Well suited for educational purposes, but simulations are very slow (does not compile into binary). Can be linked and called from R using Rnetlogo. Wilensky (1999). http://ccl.northwestern.edu/netlogo/

Swarm. It was the first platform developed for agent based simulation modelling. Initially designed in Objective-C, currently runs in Java. Well organized and stable. www.swarm.org

Box 11.2 The population: creating the individuals.

There are two general ways to define and create individuals in general purpose development environments. The methods used in specific development environments can match these or be more graphical.

Lists of objects

It consists of using a *list* to generate a collection of objects where the list refers to the population, and a *class* template of objects is used to represent agents or individuals (Box 11.3). Within the object oriented programing paradigm, classes are created to serve as templates to define objects, which in our case will refer to individuals and the properties or variables characterizing them. They can be seen as data structures. Additionally, in all languages, classes can have methods associated with them. In principle we can create our template for individuals without needing methods, using simplified class versions, if available, (e.g., *record* in *Pascal*, or *struct* in *C++*). Once we have created (declared in programming jargon) the data structure for our individuals, we need to declare and create a list to manage a collection of pointers, each of which will be used to link each individual we create. In such a manner we will be able to locate and distinguish individuals even if they have the same trait values. The list can be seen as a container that facilitates the management of individuals, allowing for adding, removing (and destroying), searching, sorting, and counting among other useful methods. In summary, we simply have to create the population (*list*) and add the number of individuals (*objects*) we need, each of them with their own set of descriptors as specified in by the conceptual model. Running many simulations can lead to problems of memory usage and allocation in the computer, depending on the environment, language and compiler. To avoid this situation we need to do the housekeeping of managing memory when destroying individuals (or any other class) and when dealing with subroutines (for example, freeing resources such as virtual memory).

Dynamic arrays

The second method consists in using dynamic arrays (arrays are simply vectors or matrices in programming jargon). Obviously they also represent a data structure in which each cell has a single value. In dynamic arrays we can keep the number of dimensions variable in run-time. Therefore, by keeping constant the dimensions characterizing the traits and variable one dimension representing the number of individuals, we can describe a population. It is easy to understand how they work by analogy with a table in a database: the columns describing trait variables will be a fixed dimension, each of which represents a trait, and each of the rows will be an individual. This second dimension will be dynamic, i.e., with a variable size because we should be able to create and delete items. Dynamic arrays also come with useful methods associated with the management of the items they contain.

Box 11.3 Pseudocode algorithm describing a basic IBM.

It represents a population with N individuals and with reproduction and survival as demographic processes. We follow the list-class approach to create the population. The model represents an exponential growth system (for example to evaluate a reintroduction in the short term or a population collapse). The explicit parameters of this model are N_0 initial population size; P_R reproduction probability; P_S survival probability; *max_age* maximum age; *t* number of time steps simulated. Note that there are other implicit parameters such as litter size, a constant that work as a model assumption. We move along all individuals of the population using conditional loops (such as *Do While*- or *For*- loops, which are sections of code that are repeated as long as a condition is met); note that we can call one subroutine from another (as for survival called from population dynamics subroutine).

```
//Declaring a container for our population, named "Population"
list Population

//Declaring the data structure for individuals (their traits)
class Individual
        Sex: string
        Age: integer

//Initializing a population of size N0;
procedure Initialize
create Population
with Population do
for 1 to N0
        create individual
        individual.sex = random(female/male)
        individual.age = random(maximum_age)
        add individual
endfor

//subroutine for reproduction with a breeding probability PR
procedure Reproduction
with Population do
N = Population size  // assign current population size to variable N
for i =1 to N do
        individual= [i]
        if individual.sex=f then
                if random<PR then
                begin
                        create individual
                        individual.sex = random(f/m)
                        individual.age = 0
                        add individual
                end
endfor

//subroutine for survival with a survival probability PS
procedure Survival
with Population do
N = Population size
for i =1 to N do
        individual= [i]
```

```
        if individual.age>max_age then delete individual else
                if random>P_S then delete individual else
                        individual.age=individual.age+1
endfor

//subroutine for population dynamics; this is the procedure we call to run the model
procedure Dynamics
N_0 = #
t = #
P_R = #
P_S = #
max_age = #
Initialize
for time = 1 to t do
        Reproduction
        Survival
        N = Population size
        plot time vs N
        save results
endfor
```

Box 11.4 Parameters, arguments and pseudorandom numbers

Parameters and arguments

With model parameters we refer to values that are relevant in our conceptual model and that need to be considered either by themselves or as part of the functional relationships. Their value can be constant in any parameterization (e.g. maximum life expectancy) or can change between parameterizations. Additionally, model parameters can be sampled from a distribution to represent not only the means but the variability of their estimates. Arguments are information that we track at run time. They are normally needed by subroutines or commands, for example, population size at a given time of a simulation, which may be required in itself as output or to calculate density. They are sometimes referred as summary statistics (Hartig *et al.,* 2011).

Parameters and arguments are stored as variables which are identified by a symbolic name (N for the argument population size or P_S for the parameter defining survival probability Box 11.3). Variables can be local or global depending on their scope. Typically we tend to use local variables when dealing with information required only within a subroutine (e.g., the variable describing the counter of a loop) and global ones when needed throughout the model. Depending on the language that we are using, variables may need to be explicitly declared, initialized, emptied before reuse and the type of information they can store needs to be defined a priori (for example, a string or an integer value). One important distinction is between variables that can hold a single value and arrays that can have multiple ordered values in one or more dimensions (i.e., vectors and matrices).

Variability and pseudorandom numbers

Some (or most) of the parameters used to parameterize a model have some associated variability in relation to both uncertainty in the empirical estimates and natural variability, typically in time, space or associated with interindividual variability. These sources of stochasticity need to be dealt with, first in the conceptual model by identifying and justifying which of them are relevant and then when defining the parameterizations that will be used for sensitivity and further analyses.

In order to obtain a stochastic value from a known distribution we use standard procedures that generate pseudorandom numbers and that are available in all compilers. These procedures need to be initialized with a seed number. If we always use the same seed, we will obtain the same sequence of numbers, which is helpful in detecting errors in the code. Typically, when running simulations we use different seeds coming from a highly variable source (such as the clock of the computer, with the help of the relevant function), thus making the sequence more unpredictable (be aware that some of the algorithms can be poor, with relatively short return rates).

Pseudorandom number generators produce numbers from a given distribution, usually a uniform distribution between 0 and 1. Unless the probability density distribution that we need is already implemented in the compiler, as often occurs with the normal distribution (with a given mean and variance that we need to specify), we can use the pseudorandom numbers obtained from the uniform distribution to randomly sample any other probability density distribution or discrete probability histogram with a bit of thought and simple math: by rejection sampling or using the inversion method (inverse transform sampling) in which we use the cumulative distribution function of the known probability distribution.

Often we may have erratic errors occurring at low rates. To locate where they occur in the code, it helps to switch off the randomization process used to generate pseudorandom numbers. In that way, the error will always occur at the very same point of the simulation, allowing you to locate the problem. We can use breakpoints in the code just before the error happens and then run the code line by line from within the compiler.

Box 11.5 Space representations

We can use two simple approaches to define space by using either a continuous or a discrete space.

Continuous space

In this case the location of each individual within the spatial domain is defined using a Cartesian or polar representation. This approach is typical of applications in which individuals move independently of an environment or at most their movement is affected only by a few spatial references that we can track with their coordinates, such as the location of other individuals or the location of a nest. The location of each individual is kept as individual traits (its coordinates) that change when it moves, whereas the spatial resolution is given by the resolution of the numeric values used (e.g., integer or floating types). Nevertheless, it is perfectly possible to use more complex vectorial map representations, which will require a bit more thinking and recalling the trigonometry we learned in secondary school

Discrete space

This approach is used in cases with more complex spatially explicit environmental properties, such as several levels of habitat quality affecting survival or movement. In that case we can represent a map as an array of one, two or three dimensions (more akin to a raster GIS landscape map), depending on the required dimensionality: one for landscapes, such as rivers, that can be represented linearly; two for x and y landscapes, and three if we need x, y and z coordinates such as in the ocean, or if using a dynamic landscape (x, y and t). In this array, each dimension is indexed between 0 and a maximum value (as defined by the domain), with the index representing the spatial location (coordinates) and the value at that location some relevant environmental property (for example, 1 for presence of a nest, 0 for absence; or different values representing different habitat qualities). The discrete space represented by the array has a typical resolution (e.g., 10x10 m or 5x5 km) which is not explicit in itself. A good way to visualize this is to think about the typical bidimensional map represented as a grid or a raster map with x and y coordinates and a stored value within each grid-cell. Grid cells can be square or take other shapes (hexagonal grids; Liu *et al.,* 1995; Letcher *et al.,* 1998). Very often the resolution of the map is also used to define the coordinates of the position of individuals, thus using only one spatial resolution in the model. If we do not use the same resolution we have to deal with the scaling between the two, the one for individuals and the one for the map, with some rules (such as rounding or truncation, behavior at the border of grid cells, etc.). For most applications grid-based approaches may be sufficient, whereas for very large domains it can be computationally demanding.

Box 11.6 ***Data in, data out***

There are three ways to parameterize a model. The simplest is by typing assigning statements in the code. For example, we can define that the variable storing the maximum age that an individual could reach equals 10 years (*max_age* = 10 in Box 11.3). This can be done with all the required information. Nevertheless, this approach is normally used with parameters that will not change in between simulations (such as constants).

If our model has a GUI, we can add components to it on which we can specify parameter values. There are many types of components, such as text, combo or drop-down list boxes, all of which have a default value that can be changed again in the form once the code is executed. Those values can easily be assigned to the relevant parameters. This method is useful to explore model behavior.

The most efficient way for the analyses is using standalone files in which we specify all the parameterization/s at once. The easiest is using text files with information delimited in some way (e.g., comma, space or tab separated values) to allow for easy identification of the values. Once the file is open and read, we can use a series of assigning statements to initialize all the variables. All this can be programmed in a subroutine which will be run early in the model to load all the parameters. Other types of files that can be used are data tables belonging to a database. This is a bit more complex since we would need to install the required ODBC (Open Database Connectivity) drivers for the specific database engine (e.g., MySQL, PostgreSQL or DB2) and some libraries in our compiler.

Retrieving output data is done in a similar way to input data: plotting graphical output in the GUI, saving it in text files or using a database engine from within the model. For example, we can add a graph component to plot the trajectory of population size (Fig. 3). Retrieving graphical output is very useful in the initial phases of model evaluation and analysis, whereas saving data in files is the standard for in-depth analyses. Keeping the output data together in the same files with the model parameters used (and the constants) is always a good recommendation to avoid future confusion.

Figure 1. Simplified scheme of the modelling cycle for model design, including the modifications that often need to be introduced during consistency checking and analyses, both in the conceptual model and its implementation in the core model and even in the way we develop the question and predictions at hand.

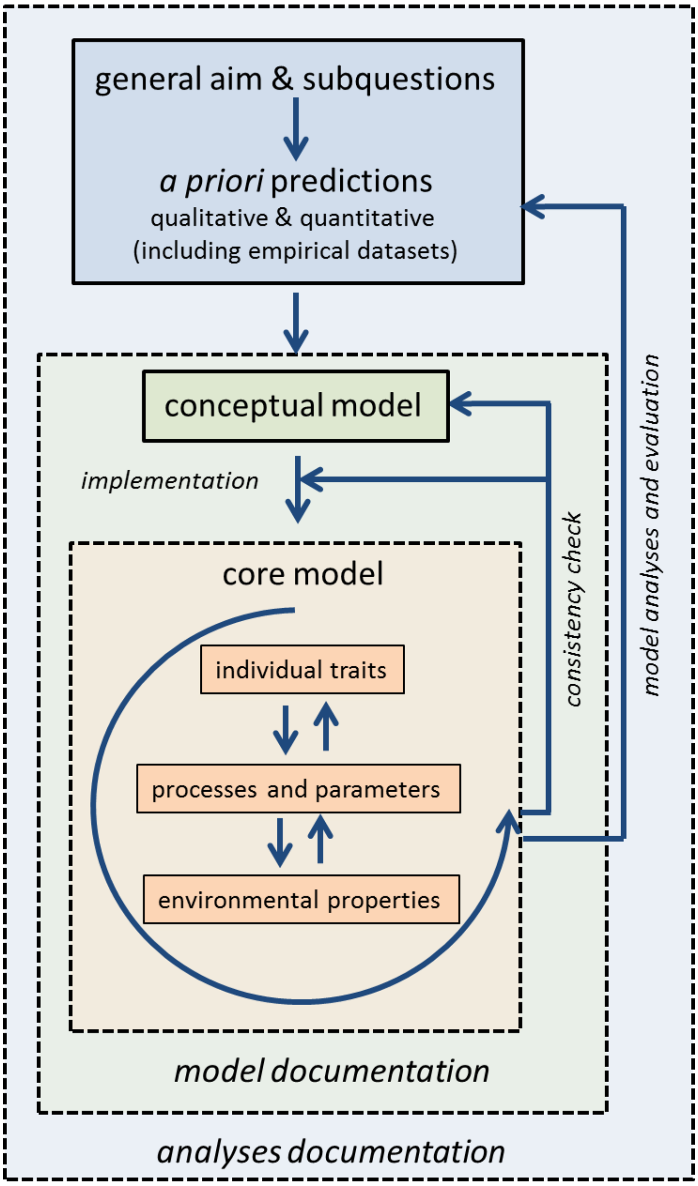

Figure 2. Schematic flow chart depicting the scheduling of a time step for the model described in Box 11.3. Time resolution is one year and space is implicit.

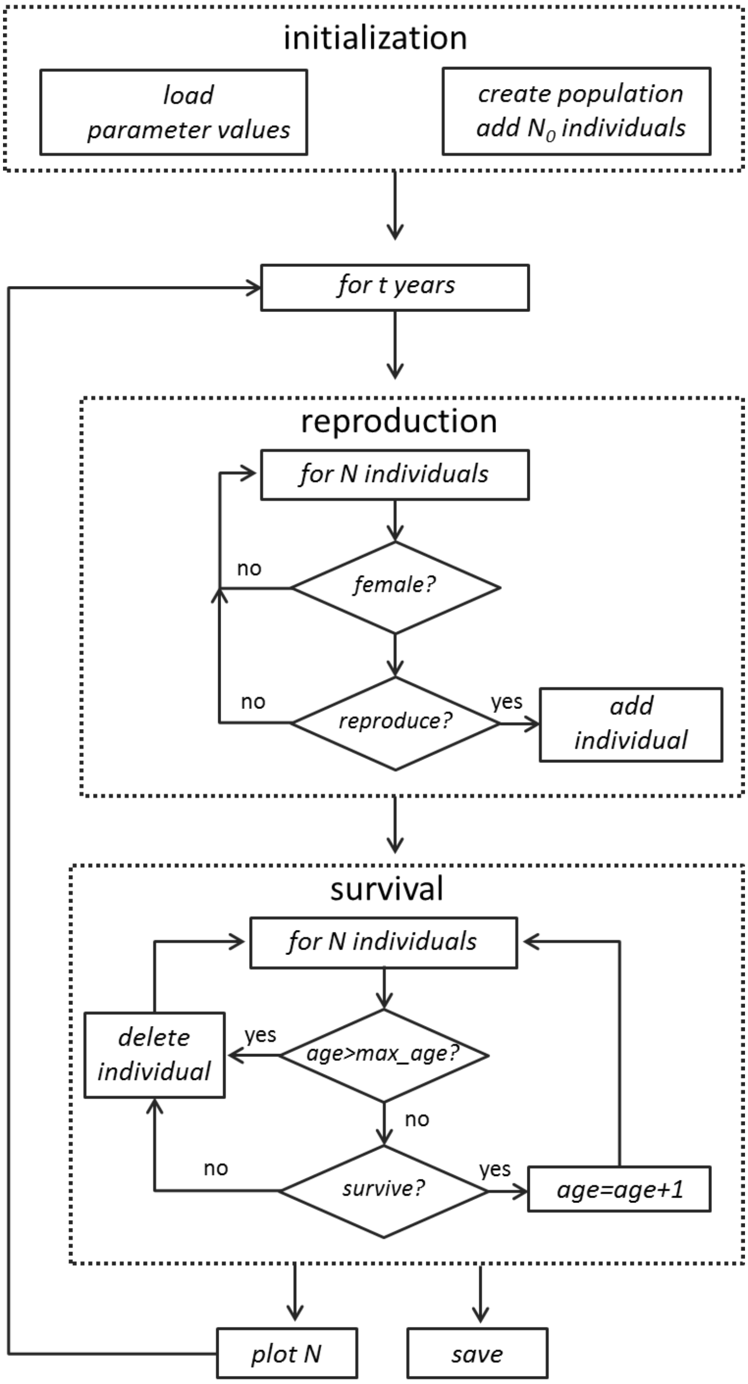

Figure 3. Graphical output for population size simulated with the model given in Box 11. 3 and parameterized with $N_0 = 30$; $P_R = 0.6$; $P_S = 0.9$; $max\_age = 10$; $t = 100$. The plot corresponds to 10 simulated population trajectories and their average (bold line). With this parameterization we observe two extinctions and the effect of the initial condition lasting for the first 15 years.

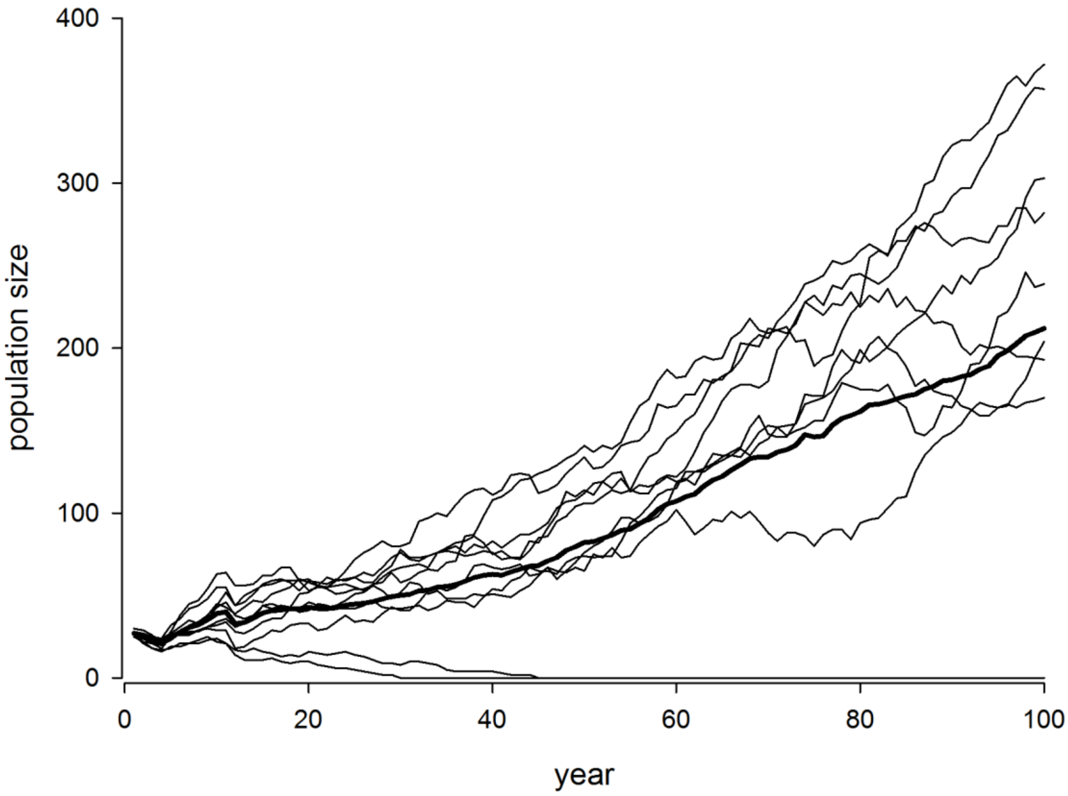

Figure 4. Schematic representation of the analyses of IBMs, including the steps of model debugging and consistency check, sensitivity and uncertainty analyses and model selection, calibration and validation. Key model predictions refer to the questions related with the questions for which the model was built. In the end, the initial questions should be answered within the inference constraints imposed by the results. Ideally, the results should help in improving the conceptual model.