Chapter 10

The Process of Modeling

Now that you are well on your way to being a modeling expert, you may begin to receive requests for assistance with various modeling projects. As a motivating example, a friend – it could also be a colleague or client – comes to you and asks for help. This friend has been involved with the effort to protect the rare Aquatic Hamster.

The Aquatic Hamster is an endangered species that spends most of its life living in lakes and rivers. Unfortunately, development and human encroachment has steadily reduced the available habitat for these hamsters and their population has plummeted. Indeed, now there is just one last population of them left located on a lake just south of the Canada/United States border.

Your friend asks you to build a model of this hamster population in order to help prioritize protection efforts and to rally support from governmental agencies and non-profits to protect this last hamster colony. You want to be of real assistance to your friend, and the hamsters are admittedly cute, so you agree to take on this modeling project.

You are at your desk to start building the model, but then realize something: You aren't sure what to do next. There are so many candidates for first steps. Do you start sketching diagrams? Do you talk to hamster experts? Do you start coding up a model? You are paralyzed by the sheer number of different choices. You know your friend is counting on you, so what do you do now?

In this chapter, we answer that question. We explore the modeling process from start to finish, introducing the tools and techniques for getting from "I need a model" to a final product that works.

Why model?

The first step to building a model is answering the simple question: *Why am I building this model?*

This question seems obvious, but it is often hard to answer in practice. Let's try answering it for our hamster population model: Why are we building this model? The truth is that so far we do not have a real understanding of this.

Oftentimes, the lack of focus begins with the friend/client/colleague who commissioned the model. They frequently do not have a strong understanding of what modeling is, including what modeling can accomplish and what it cannot. Instead, they see modeling as a means to achieve their ends, and thus are rather skimpy on the details on how these means are to come about. If you took a snapshot of what was going through their heads when they commissioned you to build the model, their thought process might look something like this:

1. Build Model.
2. . . .
3. Profit/Success/Accolades!

The missing "..." step, however, comprises the key details that no one has thought about but are what will determine success or failure of your model. The same thing is probably going on with our friend. He does not understand modeling or all the math behind models. Instead he might view a model almost as magic wand. He feels he just needs a model and then, *abracadabra*, it will solve his problem:

1. Build Model.
2. . . .
3. Hamsters Saved.

Of course this is not the case. You build a model with a specific purpose in mind otherwise it will most likely accomplish nothing. Worse yet, when it comes to the hamsters, it will be too little too late. Your first action should be to work with your friend to make sure you have filled in the "..." step. The best way to do this is generally working backwards from the final step rather than working upwards from the first one.

For us that would be to first figure out how the hamster population is to be protected. You discuss this with your friend and the two of you come to the conclusion that you will need two things in order to reliably protect the hamster population. First, government regulatory agencies need to pass (stronger) rules protecting the hamster habitat. Second, non-governmental organizations (NGO's) need to provide funds for hamster conservation and protection efforts.

Using this, we can expand our plan with more details:

1. Build Model.
2. . . .

3. Agencies enact rules to reliably protect hamsters. NGO's provide money for conservation efforts.
4. Hamsters Saved.

This focuses things for us. Rather than "Building a model to save the hamsters" (which is too vague and completely unactionable leading to our quandary about what to model), we are building a model designed to persuade governmental regulators and NGO's that they should devote resources to protecting the hamsters.

So how do we do that? Let's simplify the complex issue into two specific goals for our model:

- Show that given the *status quo* (business as usual) the hamster population will go extinct.
- Show that alternatives to the *status quo* exist (which require regulatory action or investments) that enable the hamster population not only to survive, but also to thrive.

If our model demonstrates both these things it could be a highly persuasive tool to shape decisions and policies. By building a model that does these two things[1] we will have given our friend a powerful tool to push for regulatory action and financial support.

When building your own models you'll want to go through a similar thought process to get at the core goal or question the model should address. Going into a modeling project with the attitude "First we'll build a great model, then we'll figure out how to apply it" is a prescription for failure. Of course, as you go through the process you might discover insights you never expected or you might in fact determine that your original hypothesis was wrong. Such discovery is always a great outcome, but you can never count on it happening in the course of building your model. It's best to start very focused in your modeling efforts and treat any discoveries or broadening of scope later on as a lucky bonus.

Model Project Management

When tackling modeling projects such as our hamster-population model, there are two basic overarching project management approaches. The first is founded on detailed planning and preparation. Tackling the hamster model using this approach might look something like the following sequential phases:

[1]The model of course must also inspire confidence in its audience. They must believe its results are reliable otherwise the results will have no persuasive power. Review the previous chapter for tools for building confidence in models.

Research : Find and obtain relevant literature on Aquatic Hamsters. Read peer-reviewed publications. Locate hamster experts and interview them. Identify key mechanisms affecting hamster population growth. Some mechanisms may require further study. For example, if human expansion and urbanization affect the hamster habitat area, for example, you may need to study the forces influencing urbanization. These may require additional literature searches and expert interviews.

Design : Once you have completed your background research on the hamsters, start to design the model. Create causal loop diagrams and develop stock and flow diagrams. Break your hamster population model into different sectors. You will have the hamster-specific sector, which includes sub-sectors for each of the life-stages these endangered hamsters go through. You will also need sectors for other parts of the model that affect the hamster population growth: an urbanization sector with its own model, a climate sector with a climate model, and so on. Write out equations for all these sectors and resurvey experts you have contacted to review the overall model design and the specific equations. There will probably be several cycles of iteration and model expansion during this stage as additional key areas to include are identified.

Construction : Now that you have completed a model design and received a seal of approval from experts in the field, you are ready to start building the model itself. Decide what modeling software package (or programming environment) you will use. Implement the equations as they were specified in the design phase.

Wrapping Things Up : Go through the confidence building steps from the previous chapter. Develop tests for your model to ensure it works correctly. Create model documentation. Show the model demonstrates expected behavior and obtain final approval from experts.

This approach to building a model is a very linear process where you go sequentially from stage to stage. In the project management field, this is the classic "waterfall" project where you proceed phase by phase through the project. You plan out the whole thing ahead of time estimating how long each phase would take and identifying dependencies between phases. This form of project management can work well if done expertly and it is well suited for certain kinds of projects such as constructing a building.

In our opinion, however, this approach to tackling a project is quite poorly suited to the task of building a model. There are several reasons for this.

First, each model is inherently unique[2]. You may have developed a dozen different population models in your career, but when it comes to developing a model for a new species or location, you will inevitably run into situations and problems you have never encountered before. The quantity and quality of

[2]Lots of "cookie cutter" models out there are designed to model a certain class of problems. Without custom work, however, these models are of dubious validity and may serve more to "check a box" that a model has been built rather than to be a useful decision-making tool.

data will differ from the cases before. If not, the biology of the animal you are modeling will be different. If not, the model goals and constraints will be different, and so on. Given these differences, rigid project management techniques such as the waterfall approach do not generally provide the predictability that is needed.

Secondly, when building a model you will find that many of your assumptions may simply be wrong. This can happen with every aspect of model construction: the data you thought you had will turn out to be non-existent, the equations provided to you by experts end up not working, and the model code you write will invariably have a bug or two that needs to be identified and squashed. Because of this you will continually need to be adjusting and adapting your model as you learn more about the system and what pieces of information you can rely upon and what you cannot.

Such a high likelihood of error and need for readjustment are not well suited towards techniques based on sequential, long-term planning formats. What good is a great plan if the assumptions it is based on are substantially wrong?

Take, for instance, the data you use to build your model. It is not uncommon for a collaborator to come to you and say we have X, Y and Z data series for you to use in your model (where these might represent environmental conditions or other important model inputs). When you go to check the data however you may find that in fact X does not exist (the collaborator was confused), Y actually has large gaps in the data set which make it effectively useless for your needs, and Z was collected in such a way that they were actually measuring something completely different than they thought they were.

Take, as another instance, the equations in a model. Imagine you consult an expert on Aquatic Hamsters and she provides an equation governing the survival of hamsters during their first year of life. This equation was developed as part of a scientific study where the hamsters were grown in indoor swimming pools at her university's Aquatic Hamster Research Facility. When you go to apply this equation in your model, however, you find out that how the hamsters behave when living within an indoor swimming pool is very different from how they survive in the wild. Because of this, the equation you have is simply not accurate for the hamsters living in the wild.

Errors like these two examples are *very* common. If you had proceeded with the classic waterfall approach to modeling you might not realize that you cannot rely on the data or equations you were planning to use until the very end of the modeling process. At this point it is much too late to go back and correct your model.

Failing Fast and Failing Often

Because of this, we advocate an alternative approach to building models. We support jumping right into the model construction process as early as possible.

As we showed you in the *Red* example from Chapter 4, we think it is important to get a simulation model up and running as quickly as possible. You should never want to be more than a few steps away from a simulating model.

When beginning a modeling project we recommend building the simplest model possible to get going. We call this the *Minimum Viable Model*[3] and it is the model that contains just enough to minimally represent the system and nothing more. For the hamster model, this Minimum Viable Model might contain just a single stock representing the hamster population and a couple of flows modifying the population. Nothing more.

You don't have to worry about your equations being right or your model being an accurate predictor in the Minimum Viable Model; you just want to get something up and running as soon as possible.

Once you have the Minimum Viable Model you can start to run it by people and begin to incorporate their feedback. So get your friend's thoughts on the minimal hamster model, talk to experts, study the model's forecasts and see what works and does not. Then iterate on the model: make a change here, add a new component there. If the feedback you are getting is no one trusts the model because it does not contain some key mechanism, add that mechanism to the model[4]. Steadily adjust and refine the model based on the actual results of the model and the feedback you receive.

This feedback will be more useful to you when you have a concrete model that is simulating than it would be if you were just running abstract ideas by people. By putting your stake in the ground with a model that simulates, you allow others to critique and engage with the model providing you with valuable information about what works and what does not. If you do not come with a concrete model, you run the risk of receiving very vague, unactionable feedback.

What is best about this approach of rapid iteration we advocate is that it allows you to identify failures quickly. If a data source is no good, you find that out immediately as you try to integrate it rather than spending days, weeks or months planning your model with the assumption it's really there or you can really use it. Failing fast and failing often is a key goal in the model development process. Your successes in life are generally directly proportional to the number of failures you experience; and the same is true in modeling. By speeding up the process of identifying and iterating past failures, this agile

[3]This idea is adapted from Eric Ries's excellent book *The Lean Startup* (@Ries:2011vp). In it he advocates an approach to developing start-up companies and businesses focus on rapid development and innovation. Ries supports developing a "Minimal Viable Product" for the company as quickly as possible and iterating on the feedback received for this initial product.

[4]But the key is to wait until you get this feedback. It's easy on your own or with a group of people to make a list of dozens mechanisms that a model *must* contain to be realistic. Once you have implemented those mechanisms in your model you might find out that no one actually cared about them. It is best to start small and then augment the model when there is a demand for some additional mechanism, than it is to spend a large amount of time implementing a very complex model and then to find out much of that work was unnecessary.

approach to modeling will often result in higher quality models completed more quickly than approaches that rely on extensive planning.

From Mental Models to Simulation Models

Generally speaking, a single individual should ultimately be responsible for the design and implementation of a model. Models "designed-by-committee" are understandably suffused with compromise and a greater lack of focus. That said, even though one person is ultimately calling the shots, many voices and perspectives are there to be heard in the modeling process. The more input there is into a model, the better the resulting model will most likely be.

The people you are working with generally will not be experts in modeling. Because of this, even if they are intimately familiar with the system you are attempting to model, it will sometimes be difficult to take their freeform insights and transform them into a formal model structure and accompanying numerical equations. In fact people often have great difficulty communicating and describing their own mental models of a system. A number of useful tools and techniques can be used to help elicit information on people's mental models. We discuss three of these tools in the following sections.

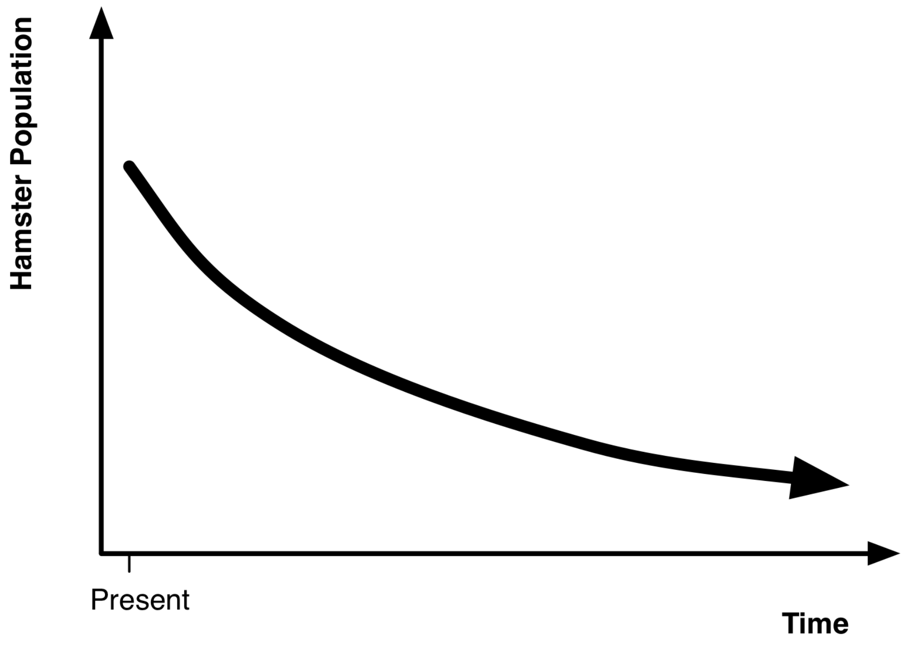
Reference Mode Diagrams

A reference mode is a graph that plots how the key stocks and variables in the system change over time. The *x*-axis of the graph is time, and the *y*-axis shows the values of the variables as they change. Sometimes reference modes are based on historical data, but you can also create them by asking those involved with the system to sketch out how they think the system will behave in different scenarios.

For our hamster model we could start simply by asking our friend to sketch out what he thinks will happen with the hamster population in the future assuming business as usual (remember that the status quo does not mean no-action). When we do this, he sketches out the top graph in Figure 1.

While your friend probably would use different terminology, to us the curve he sketched immediately looks like an exponential decay model. The instant we see this sketch we should start mapping out a stock and flow diagram in our mind to implement this type of model. Your friend does not need to understand any modeling concepts though, he just needs to be able to draw a picture of what he thinks will happen in the future. This is something that is easy to ask most people to do.

Let's go beyond the simple business as usual scenario. We can also use reference mode diagrams to elicit information on different scenarios. For instance, we have previously been told that development and encroachment on the hamster habitat are key factors reducing the hamster population size. Not only does the



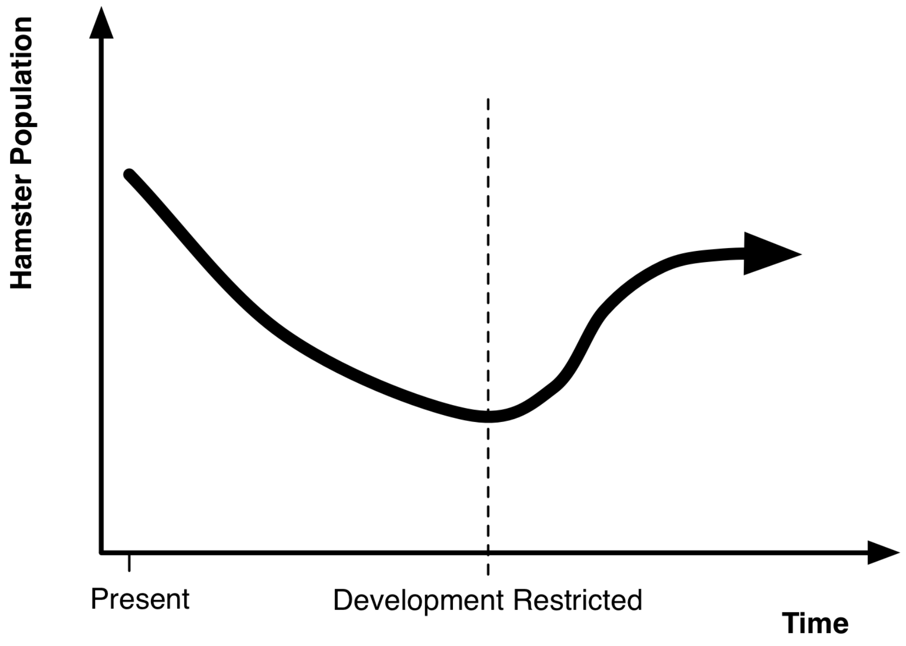


Figure 1. Sample reference modes for our hamster model.

development consume key hamster habitat, the construction creates disturbances that have a further negative impact on the hamsters.

We can ask our friend to create a second sketch that shows how the hamster population would respond if development were suspended indefinitely. He responds by drawing the bottom graph in Figure 1. This graph shows the hamster population starting to recover after development stops, initially growing and then leveling off at a certain point.

Again, your friend never said this, but looking at this second drawing we should immediately start thinking of logistic growth models. The leveling off implies that there is some carrying capacity limit for the hamsters. This carrying capacity is probably a function of the available hamster habitat and the disturbances that are going on around the hamsters. We can start to sketch out stock and flow and causal loop diagrams to implement these types of dynamics and reproduce the behaviors our friend has drawn.

These are just two of the reference modes we might ask our friend to think about. We could go on to explore other scenarios and see how he thinks the changes in the scenarios would affect the hamster population. We could also ask him to sketch out other key variables in the system – such as the quantity of food available to the hamsters – to understand how he thinks these key variables interact. We could go on to interview other people familiar with the system and take them through the same process. Ideally, all the reference modes between individual people will agree, but differences are in themselves also useful in revealing different mental models between our interviewees. Bridging differences will be a key interest of ours as we attempt to develop a persuasive model that will bring everyone on board and gain wide support.

Asking non-modelers to sketch out reference modes is a great technique for several reasons. Reference modes are accessible to laypeople, force your interviewees to be concrete, and provide you with very useful and actionable material. Really, a reference mode is a projection of an individual's mental model of the system. They may be unable (or unwilling) to explain their mental model to you in equations or even words, but they generally will be able to describe how they perceive the world using these reference mode diagrams – one small slice of their mental model at a time. Once you have the diagrams, you can proceed to translate them into model structure and equations.

Pattern-Oriented Modeling

Pattern-oriented modeling focuses on identifying key patterns in the system to be modeled. For example, we may observe a boom-and-bust pattern in our hamster population that is triggered by unusually warm weather. When we develop our model, we formulate relationships and equations that will replicate this boom-and-bust pattern in the simulation.

Developed to help guide the creation of agent-based models, pattern-oriented modeling is very similar in concept to reference modes and system archetypes.

Rather than building models around expected dynamic trajectories however, pattern-oriented modeling builds models to recreate patterns. Sometimes a pattern may be the same as a reference mode, but especially when dealing with agent based modeling you may not be able to define a pattern in terms of the dynamic trajectory of a reference mode. For a good overview of pattern-oriented modeling, see @Grimm:2005ei.

Group Model Building

Group modeling sessions are a powerful tool to capitalize on the collective thoughts of a group to inform model structure and design. Instead of individually surveying experts and those involved in a system, a group session with many interested parties can be conducted. The term "group model building" is a bit of a misnomer as generally the model itself will be built away from the group by the facilitator or modeler and the group work will be focused on identifying and ranking key variables and mechanisms and developing high-level causal loop or stock and flow diagrams. See @Andersena:1997tg for a very practical overview of running and facilitating group model building sessions.

Group modeling sessions can also benefit an organization independently of the success or failure of the model itself. You might expect that the mental models of individuals within an organization would be aligned and the members of the organization would share a common objective and understanding of the challenges and needs required to achieve this objective. However, this is often not the case as different organization members may hold distinct mental models of the organization's purpose and operation within the world. Additionally, it is quite possible that these differences may never be realized as people may fail to adequately communicate their mental models assumptions and beliefs during the course of regular interactions.

The group modeling process can force the concrete discussion of and revealing of these mental models and the stakes involved in having these differences. Once they are revealed, they can be discussed and reconciled, potentially leading to a greater congruity of viewpoints within the group and a greater shared purpose. @Vennix:1993wv carried out a survey of participants in group model building sessions and found that this process led to insights and a shared vision more quickly than occurred in standard meetings.

Wrapping it Up

Completing a model is in some ways just the first step in a modeler's work. Once the model is finished you need to make sure to develop adequate tests to ensure it is operating as designed. Moreover, a model by itself is often of little use. You will need to develop extensive sets of documentation, manuals and tutorials if you want the model to be used in practice by people other than yourself. Such efforts take time. Writing clear and useful documentation is a skill in itself and, if done right, may take as long as developing the model in the first place!

Chapter 14

Optimization and Complexity

We start this chapter by taking up our hamster population model from earlier and reconsidering it. As you recall, your friend requested our help in constructing a model to simulate the population of the endangered Aquatic Hamsters. There are many ways to exploit valuable empirical data to improve your models. For instance, if we had data on hamster fecundity, we might be able to plug that information in directly as a parameter in our hamster population model.

One of the most useful kinds of empirical data is historical time series. Some of these time series might represent data and factors that feed into the model, but are not directly modeled. For example, we might have historical temperature data. The temperature could be an important thing to include in the model, as it would affect hamster survival however it is not something we directly model. By this we mean that we do not expect our hamsters to have any effect on the temperature in the region but we do expect the temperature to have an effect on the hamsters. Thus, we can feed the temperature data into the model. We can do this by importing this historical temperature data and including it in the model using a converter primitive.

In other cases, the historical data may represent factors you are directly trying to model. For example, we have a data series of biannual hamster population surveys going back 20 years. This data series lets us know roughly how many hamsters there were over time. Because we are trying to model this data, it is not something plug directly into our model as we could with the temperature, but it is something we can use to calibrate and assess the accuracy of our model.

How do we do this and what will be the results?

Assessing Model Accuracy

We first import our historical data into a converter primitive. We then assess the accuracy of the model in two ways: qualitatively and quantitatively. To assess how well our model fits the historical data qualitatively we plot the simulated and historical data series next to each other. Ideally, they will match up closely but if they do not we should pay close attention to how they differ.

If they have the same general shape (except for a vertical or horizontal displacement) that is good news, as it indicates that you may have gotten the general dynamics of your model correct and that you may just need to fine-tune the relationships and parameter values. If the results look considerably different you may have more work to do in improving the model.

You can also assess the accuracy of models quantitatively. One standard tool people use to assess the accuracy of a model is the R^2 metric[1]. R^2 is the fraction of the squared error explained by the model compared to the "null" model. It ranges from 0 (the model basically provides no predictive power), to 1 (the model predicts perfectly). Mathematically, R^2 is calculated like so:

$$R^2 = \sum_t \frac{(\overline{\text{Truth}} - \text{Truth})^2 - (\text{Model} - \text{Truth})^2}{(\overline{\text{Truth}} - \text{Truth})^2}$$

Naively used, R^2 has a number of issues that we will discuss later in this chapter. However, it is still a useful tool that many people use and with which they are familiar. It is also relatively straightforward to calculate. The following code calculates an R^2 for a model fit. This is code written in JavaScript and can be placed as the **Action** for a button primitive in Insight Maker. The code is written assuming two primitives: a converter [**Historical Hamsters**] containing historical population sizes and a stock [**Hamsters**] containing simulated population sizes. You can edit the code to reference the actual names of the primitives in your model.

```
var simulated = findName("Hamsters"); // Replace with your primitive name
var historical = findName("Historical Hamsters"); // Replace with your primitive name

var results = runModel({silent: true});

var sum = 0;
for(var t = 0; t < results.periods; t++){
    sum += results.value(historical)[t];
}

var average = sum/results.periods;

var nullError = 0;
var simulatedError = 0;
for(var t = 0; t < results.periods; t++){
```

[1]Though this metric is not often used in systems dynamics or agent-based models, it is widely used for statistical models such as linear regressions.

```
    nullError += Math.pow(results.value(historical)[t] - average, 2);
    simulatedError += Math.pow(results.value(historical)[t] - results.value(simulated)[t], 2)
}

showMessage("Pseudo R^2: "+((nullError-simulatedError)/nullError));
```

Calibrating the Model

In addition to using historical data to assess the model fit, you can also use historical data to calibrate model parameters. Depending on the model, you may have many parameters for which you do not have a good way to determine their values. Earlier, we discussed how to use sensitivity testing to assess whether our results are resilient to this uncertainty and to build confidence in the model. Another way to build confidence in your parameter values is, instead of guessing the values of these uncertain parameters, to choose the set of values that results in the best fit between simulated and historical data. This is a semi-objective criterion that helps to remove personal biases you might have from the modeling process.

Goodness of Fit

The first step to using historical data to calibrate the model parameters is to understand what is meant by "the best fit" between historical and simulated data. Conceptually, the idea of a "good fit" seems obvious. A good fit is one where the historical and simulated results are very close together (a *perfect* fit is when they are the same, but that is generally more than we can hope for). However, putting a precise mathematical definition on the concept is not trivial.

Many commonly used goodness of fit measures exist, and below we list some key ones.

Squared Error

Squared error is probably the most widely used of all measures of fit[2]. To calculate the squared error we carry out the following procedure. For each time period we take the difference between the historical data value and the simulated value and then we square that difference. We then sum up all these differences to obtain the total error for the fit. Higher totals indicate worse fits, and lower totals indicate better fits.

The following equation could be placed in a variable to calculate the squared error between a primitive named [**Simulated**] and one named [**Historical**]:

```
([Simulated]-[Historical])^2
```

[2] The key reason for this is that regular linear regression (ordinary least squares, the most widely used modeling tool) uses squared error as its measure of goodness of fit. Doing so simplifies the mathematics of the regression problem greatly in the linear case.

Please note that maximizing the R^2 measure we described earlier is equivalent to minimizing the squared error.

Absolute Value Error

A characteristic of squared error is that outliers have high penalties compared to other data points. Outliers are points in time where the fit is unusually bad. Since the squared error metric squares the differences between simulated and historical data, large differences can cause even larger amounts of error when they are squared. This can sometimes be a negative feature of squared error if you do not want to outliers to have special prominence and weight in the analysis.

An alternative to squared error that treats all types of differences the same is the absolute value error. Here, the absolute value of the difference between the simulated and historical data series is taken. The following equation could be placed in a variable to calculate the absolute value error between a primitive named [**Simulated**] and one named [**Historical**]:

```
Abs([Simulated]-[Historical])
```

Other Approaches

Many other techniques are available for measuring error or assessing goodness of fit. Most statistical approaches function by specifying a full probability model for the data and then taking the goodness of fit not as a measure of error, but rather as the *likelihood*[3] of observing the results we saw given the parameter values. To be clear the issue of optimizing parameter values for models is one that is more complex than what we have presented here. Many sources of error exist in time series and analyzing them is a very complex, statistical challenge. The basic techniques we have presented are, however, useful tools that serve as gateways towards further analytical work.

Finding the Best Fit

After choosing how to measure the quality of a fit quantitatively, we need to find the set of parameter values that maximize the fit and minimize the error. To do this we use a computer algorithm called an optimizer that automatically experiments with many different combinations of parameter values to find the set of parameters that has the best fit.

Many optimizers basically work by starting with an initial combination of parameter values and measuring the error for that combination. The optimizer then slightly changes the parameter values in order to check the error at nearby combinations of parameter values. For instance, if you are optimizing one

[3]Likelihood is a technical statistical term. It can be roughly thought of as equivalent to "probability", though it is not precisely that.

parameter, say the hamster birth rate, and your initial starting value is a birth rate of 20% per year; the optimizer will first measure the error at 20% and then measure the errors at 19% and 21%.

If one of the neighbors has a lower error than the initial starting point, the optimizer will keep testing additional values in that direction. It will steadily "move" towards the combination of parameters that results in the lowest error, one step at a time. If, however, the optimizer does not find any nearby combination of parameter values with a lower error than its current combination of parameter values, it will assume it has found the optimal combination of parameter values and stop searching for anything better.

The precise details of optimization algorithms are not important. You need to be aware of one key thing however: these algorithms are not perfect and they sometimes make mistakes. The root cause of these mistakes are so-called "local minimums". An optimizer works by searching through combinations of different parameter values trying to find the combination that minimizes the error of the fit. The combination that has the smallest error out of all possible combinations is known as the true minimum or the "global" minimum.

A local minimum is a combination of parameter values that are not the global minimum, yet whose nearby neighbors all have higher errors. Figure 1 illustrates the problem of local minimum. If the optimizer starts near the first minimum in this figure it might head towards that minimum without ever realizing that another, improved minimum exists. Thus, if you are not careful, you may think you have found the optimal set of parameters when in fact you have only found a local minimum that might have much worse error than the true minimum.

! Figure 1. An illustration of local and global minimum for an optimization problem involving a single parameter.

There is no foolproof way to deal with local minimums and no guarantee that you have found the true minimum[4]. The primary method for attempting to prevent an optimization from settling in on a local minimum is to introduce stochasticity into the optimization algorithm. Optimization techniques such as *Simulated Annealing* or *Genetic Algorithms* will sometimes choose combinations of parameter values at random that are actually *worse* than what the optimizer has already found. By occasionally moving in the "wrong" direction, away from the nearest local minimum, these optimization algorithms are more resilient and less likely to become stuck on a local minimum and more likely to keep searching for the global minimum.

Unfortunately, in our experience we have not been satisfied by the performance of these types of stochastic optimization algorithms. They are generally very slow and without fine-tuning by an expert can still easily become stuck in a

[4]This is true for the type of optimization problems you will generally be dealing with. Other types of optimization problems are much easier than the ones you may be encountering, as they are what are known as *convex* optimization problems and are guaranteed not to have any local minimums.

local minimum. We prefer to use non-stochastic deterministic methods as the core of our optimizations. We then introduce stochasticity into the algorithm by using multiple random starting sets of parameter values. For instance, instead of carrying out a single optimization we will do 10 different optimizations each starting at a different set of parameter values. If all 10 optimizations arrive at the same final minimum that is strong evidence we have found the global minimum. If they all arrive at different minima, then there is a good chance we have not found the global minimum.

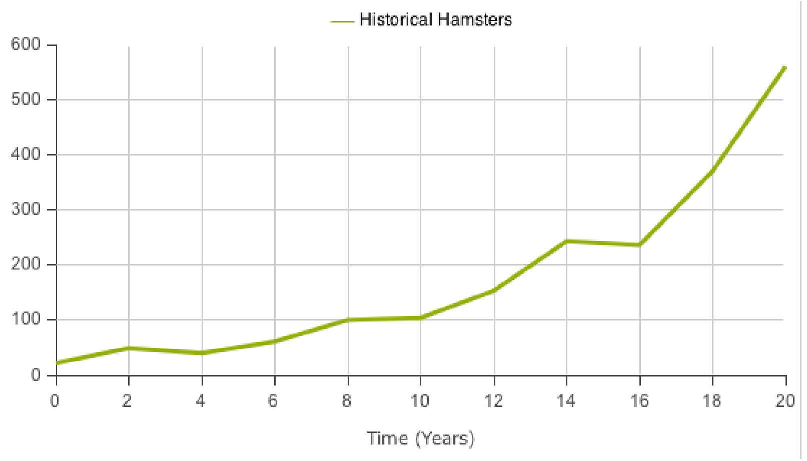
Optimizing Parameter Values

This model illustrates the use of optimization and historical data to select the growth rate for a simulated population of hamsters.

1. Create a new **Converter** named [**Historical Hamsters**].
2. Change the **Data** property of the primitive [**Historical Hamsters**] to `0, 22; 2, 49; 4, 40; 6, 61; 8, 100; 10, 104; 12, 153; 14, 243; 16, 236; 18, 370; 20, 560`.
3. The model diagram should now look something like this:

Historical Hamsters

4. We start by importing our historical population data into a converter primitive. In this illustrative example we have twenty years of data with a census of the hamster population being carried out every two years. We run the model to see what this historical data looks like.
5. Run the model. Here are sample results:



6. There is a lot of variability and the population even declines some years. However, it looks like in general the rate of growth increases as the population size increases. This is what we would expect to see with exponential growth. Let's build a simple exponential growth model to attempt to replicate what we see with the historical data.

7. Create a new **Stock** named [**Hamsters**].

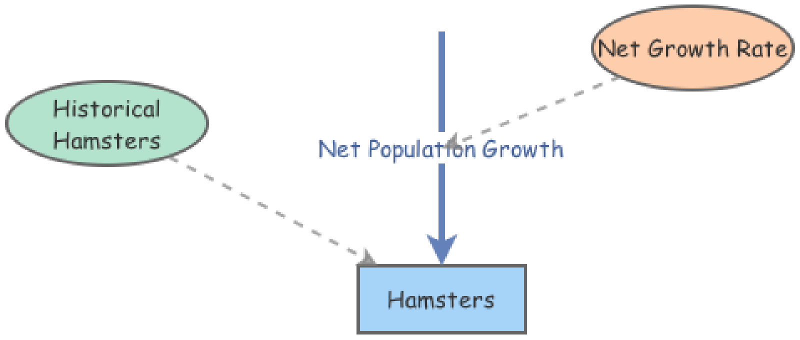
8. Create a new **Flow** going from empty space to the primitive [**Hamsters**]. Name that flow [**Net Population Growth**].

9. Create a new **Variable** named [**Net Growth Rate**].

10. Create a new **Link** going from the primitive [**Net Growth Rate**] to the primitive [**Net Population Growth**].

11. Create a new **Link** going from the primitive [**Historical Hamsters**] to the primitive [**Hamsters**].

12. The model diagram should now look something like this:



13. That's the structure of our model. Now we can fill in the equations. We'll set the initial population size for our simulated hamster population to be the same as for the historical data.

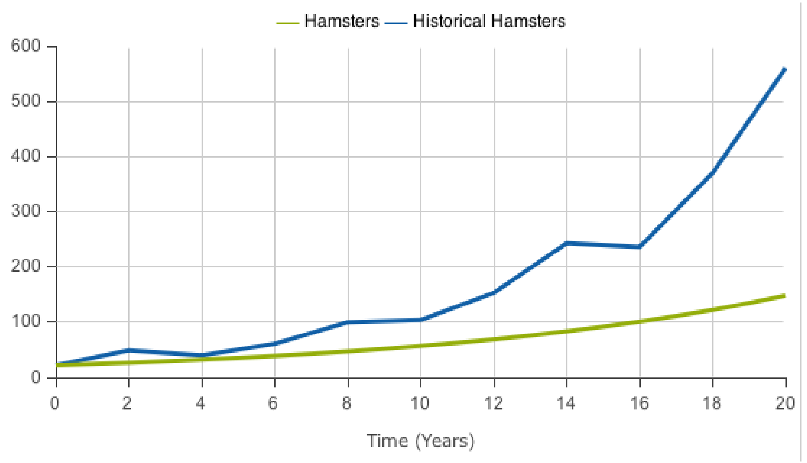
14. Change the **Initial Value** property of the primitive [**Hamsters**] to `[Historical Hamsters]`.

15. Change the **Flow Rate** property of the primitive [**Net Population Growth**] to `[Hamsters]*[Net Growth Rate]`.

16. What growth rate should we begin with? We do not have any data on this. Let's experiment by starting with 10% per year and see what we end up.

17. Change the **Equation** property of the primitive [**Net Growth Rate**] to `0.1`.

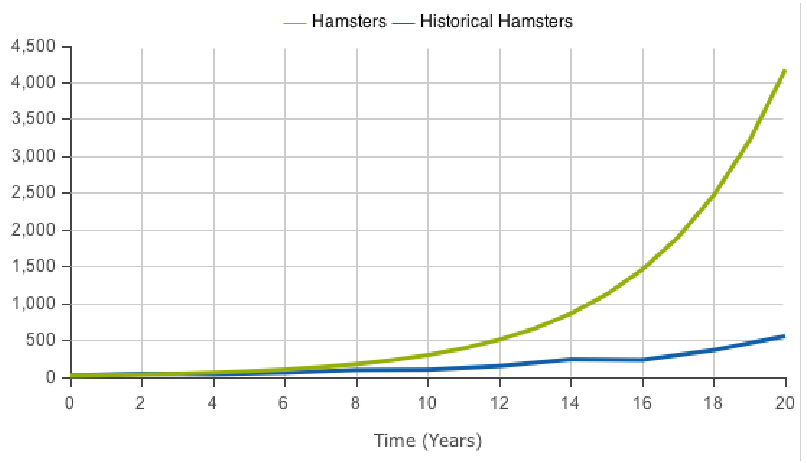
18. Run the model. Here are sample results:



19. That does not look too great. Our simulated population is much smaller than the historical values. Let's try a larger growth rate, say 30%.

20. Change the **Equation** property of the primitive [**Net Growth Rate**] to `0.3`.

21. Run the model. Here are sample results:



22. That's not good either, now our population is too large! We could keep experimenting with different growth rates to find a good one, but that might take a while. Let's just let the optimizer do the work for us. First we need to create a primitive to hold the error. We will use the squared error measure we discussed earlier.

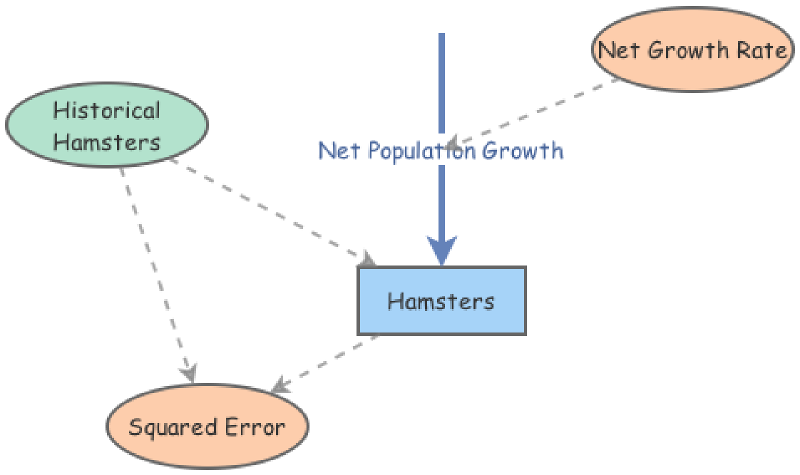
23. Create a new **Variable** named [**Squared Error**].

24. Create a new **Link** going from the primitive [**Hamsters**] to the primitive [**Squared Error**].

25. Create a new **Link** going from the primitive [**Historical Hamsters**] to the primitive [**Squared Error**].

26. Change the **Equation** property of the primitive [**Squared Error**] to `([Hamsters]-[Historical Hamsters])^2`.

27. The model diagram should now look something like this:

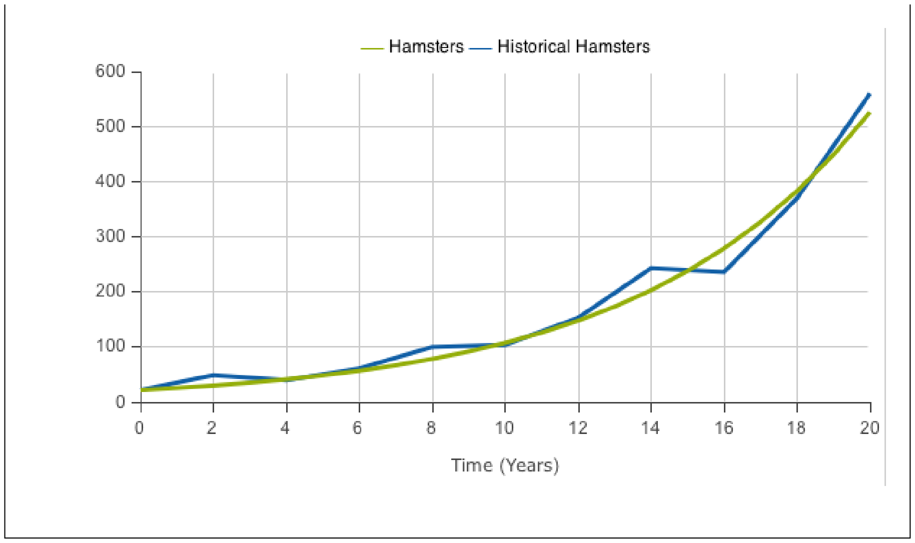


28. There, we have set up what we need for the optimizer to work. Now we can run the optimizer. We set the **Goal Primitive** to [**Squared Error**] and the **Primitive to Change** to [**Net Growth Rate**]. We tell the optimizer to minimize the integral of the error and set the optimizer to work.

29. The optimizer gets our results almost instantly: 0.172 or 17.2% is the optimal growth rate. When we run the model with this value the results look great. It's an almost perfect match between the historical and simulated data.

30. Change the **Equation** property of the primitive [**Net Growth Rate**] to `0.172`.

31. Run the model. Here are sample results:



Exercises

Calculate the pseudo R^2 for [**Growth Rate**] = `0.1`, `0.3`, and `0.172`.

The Cost of Complexity

After a good deal of work and many sleepless nights you have completed the first draft of your Aquatic Hamster population model. The results are looking great and your friend is really impressed. When he runs it by some colleagues however, they point out that your model does not account for the effects of the annual Pink Spotted Blue Jay migration.

Pink Spotted Blue Jays (PSBJ) are a species of bird that migrates every fall from northern Canada to Florida. In the spring they return from Florida to Canada. Along the way, they usually spend a few days by the lake where the Aquatic Hamsters have their last colony. During this time they eat the same Orange Hippo Toads the hamsters themselves depend upon as food. By reducing the Hippo Toad population, the PSBJ negatively affect the hamsters, at least for this period of time when there is less food available to support them.

The timing of the PSBJ migration can vary by several weeks each year no one knows precisely when the PSBJ's will arrive at the lake or even how long they will stay there. Further, the population of migrating birds can fluctuate significantly with maybe 100 birds arriving one year and 10,000 another year. The amount of toads they eat is proportional to the number of birds. Not much data exist quantifying the birds' effects on the hamsters, but it is a well-established fact that they eat the Hippo Toads the hamsters rely upon for their survival and many conservationists are concerned about the migration.

Your friend's colleagues wonder why you have decided to not include the PSBJ migration in your model. They want to know how they can trust a model that does not include this factor that clearly has an effect on the hamster population.

In response, you may point out that though the migration clearly has an impact, it appears to be a small one that is not as important as the other factors in the model. You add that there are no scientific studies or theoretical basis to define exactly how the migration functions or how it affects the hamster population. Given this, you think it is probably best to leave it out.

You say all this, but they remain unconvinced. "If there is a known process that affects the hamster population, it should be included in the model," they persist. "How can you tell us we shouldn't use what we know to be true in the model? We know the migration matters, and so it needs to be in there."

The Argument for Complexity

Your friend's colleagues have a point. If you intentionally leave out known true mechanisms from the model, how can you ask others to have confidence that the model is accurate? Put another way, by leaving out these mechanisms you ensure the model is wrong. Wouldn't the model *have* to be better if you included them?

This argument is, on the surface, quite persuasive. It is an argument that innately makes sense and appeals to our basic understanding of the world: Really it seems to be "common sense".

It is also an argument that is wrong and very dangerous.

Before we take apart this common sense argument piece by piece, let us talk about when complexity is a good thing. As we will show, complexity is not good from a modeling standpoint, but it can sometimes be a very good tool to help build confidence in your model and to gain support for the model.

Take the case of the PSBJ migration. It might be that adding a migration component to the model ends up *not* improving the predictive accuracy of the model. However, if other people view this migration as important, you may want to include the migration in the model if for no other reason than to get them on board. Yes, from a purely "prediction" standpoint it might be a waste of time and resources to augment the model with this component, but this is sometimes the cost of gaining support for a model. A "big tent" type model that brings lots of people on board might not be as objectively good as a tightly focused model, but if it can gain more support and adoption it might be able to effect greater positive change.

The Argument Against Complexity

Generally speaking, the costs of complexity to modeling are threefold. Two of them are self evident: there are computational costs to complex models as they take longer to simulate and there are also cognitive costs to complex models in that they are harder to understand. There is, however, a third cost to complexity that most people do not initially consider: complexity often leads to less accurate models compared to simpler models.

In the following sections we detail each of these three costs.

Computational Performance Costs

As a model becomes more complex, it takes longer to simulate. When you start building a model it may take less than a second to complete a simulation. As the model's complexity grows, the time required to complete a simulation may grow to a few seconds to a few minutes and then to even a few hours or more.

Lengthy simulation times can significantly impede model construction and validation. The agile approach to model development we recommend is predicated on rapid iteration and experimentation. As your simulation times cross beyond even something as small as 30 seconds, model results will no longer be effectively immediate and your ability to rapidly iterate and experiment will be diminished.

Furthermore, when working with an optimizer or sensitivity-testing tool, performance impacts can have an even larger effect. An optimization or sensitivity testing tool may run the model thousands of times or more in its analysis so even a small increase in the computation time for a single simulation may have a dramatic impact when using these tools.

Optimizations themselves are not only affected by the length of a simulation, they are also highly sensitive to the *number* of parameters being optimized. You should be extremely careful about increasing model complexity if this requires the optimizer to adjust additional parameter values. A simplistic, but useful, rule of thumb is that for every parameter you add for an optimizer to optimize, the optimization will take 10 times as long[5].

Thus if it takes one minute to find the optimal value for one parameter, then it takes 10 minutes to find the optimal values for two parameters and 100 minutes to find the optimal values for three parameters. Imagine we had built a model and optimized five parameters at once. We then increased the model complexity so we now had to optimize ten parameters. Our intuition would be that the optimization would now take twice as long. This is wrong. Using our power of ten rule we know that the time needed will be closer to 10^5 or 100,000 times as long!

That is a huge difference and highlights how important it is to keep model complexity at a manageable level. In practice, a rule of thumb is that you should

[5]In practice an optimizer should ideally perform a bit better than this, but this provides us a useful guideline to understand optimizations. Also it should be noted that the optimizations we are talking about here are for non-linear optimization problems for which gradients (derivatives) cannot be directly calculated. For other types of optimization problems, such as linear problems, much faster optimization techniques are available.

have no difficulty optimizing one or two parameters at a time. As you add more parameters that optimization task becomes rapidly more difficult. At five or so parameters you have a very difficult but generally tractable optimization challenge. Above five parameters you may be lucky to obtain good results.

Cognitive Costs

In addition to the computational cost of complexity, there is also a cognitive cost. As humans we have a finite ability to understand systems and complexity. This is partly why we model in the first place: to help us simplify and understand a world that is beyond our cognitive capacity.

Returning to our hamster population model, including the bird migration could create a confounding factor in the model that makes it more difficult to interpret the effects of the different components of the model and extract insights from them. If we observe an interesting behavior in the expanded model we will have to do extra work to determine if it is due to the migration or some other part of the model. Furthermore, the migration may obscure interesting dynamics in the model making it more difficult for us to understand the key dynamics in the hamster system and extract insights from the model.

We can describe this phenomenon using a simple conceptual model defined by three equations. The number of available insights in a model is directly proportional to model complexity. As the model complexity increases, the number of insights available in the model also grows.

$$\text{Available Insights} \propto \text{Complexity}$$

Conversely, our ability to understand the model and extract insights from it is inversely proportional to model complexity:

$$\text{Understandibility} \propto \frac{1}{\text{Complexity}}$$

The number of insights we actually obtain from a model is the product of the number of available insights and our ability to understand the model:

$$\text{Insights} = \text{Available Insights} \times \text{Understandability}$$

Thus when the model complexity is 0 – in effect basically no model – we gain no insights from the model. As the model complexity starts to rise, we begin to gain additional insights. After a certain point however, the added model complexity actually inhibits additional understanding. As complexity rises our insights will fall back down towards 0. This phenomenon is illustrated in Figure 2.

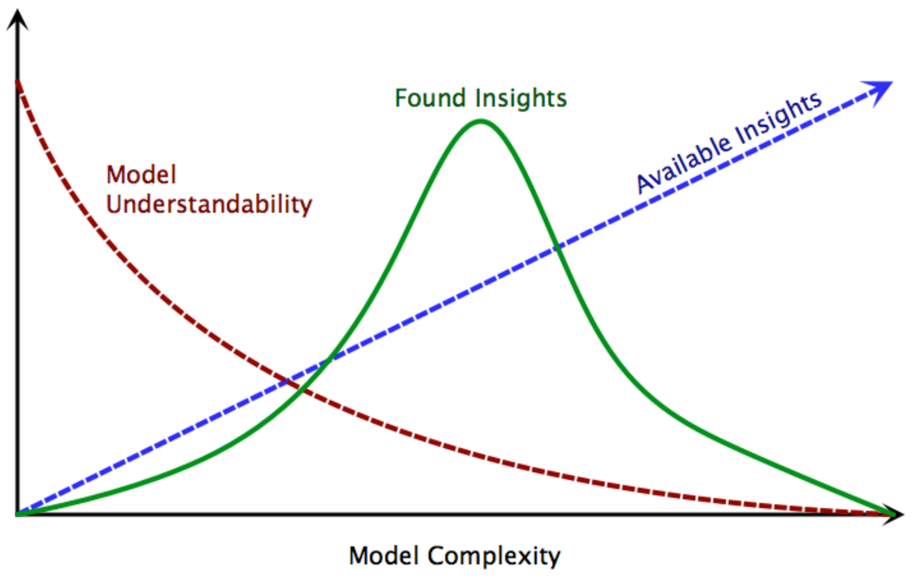


Figure 2. Expected discoveries of insights as model complexity increases.

Accuracy Costs

The negative effects of complexity on computational performance and our cognitive capacity should not be a surprise. What may be surprising on the other hand, is the fact that complex models are in fact often *less accurate* than simpler alternatives.

To illustrate this phenomenon, let us imagine that for part of our hamster population model we wanted to predict the size of the hamsters after a year[6]. The hamsters go through two distinct life stages in their first year: an infant life stage that lasts 3 months and a juvenile life stage that lasts 9 months. The hamsters' growth patterns are different during each of these periods.

Say a scientific study was conducted measuring the sizes of 10 hamsters at birth, at 3 months and at 12 months. The measurements at birth and 12 months are known to be very accurate (with just a small amount of error due to the highly accurate scale used to weigh the hamsters). Unfortunately, the accurate scale was broken when the hamsters were weighed at 3 months and a less accurate scale was used instead for that period. The data we obtain from this study are tabulated below and plotted in Figure 3:

Hamster	Birth	3 Months	12 Months

[6]Size could affect hamster survival and fecundity so it could be an important variable to model.

1	9.0	23.2	44.4
2	9.7	19.8	44.0
3	10.2	23.5	44.7
4	8.8	32.2	43.3
5	10.1	31.3	44.5
6	10.0	27.2	44.2
7	10.0	21.4	46.1
8	11.1	24.1	46.0
9	8.7	41.0	44.9
10	11.2	31.7	43.8

Now, unbeknownst to us, there are a pair of very simple equations that govern Aquatic Hamster growth. During the infant stage they gain 200% of their birth weight in that three-month period. Their growth rate slows down once they reach the juvenile stage such that at the end of the juvenile stage their weight is 50% greater than it was when they completed the infant stage. Figure 3 plots this true (albeit unknown) size trajectory compared to the measured values. The higher inaccuracy of the measurements at 3 months compared to 0 and 12 months is readily visible in this figure by the greater spread of measurements around the 3 month period.

We can summarize this relationship mathematically:

$$\text{Size}_{t=3\text{ months}} = 3.00 * \text{Size}_{t=0\text{ months}}$$
$$\text{Size}_{t=12\text{ months}} = 1.50 * \text{Size}_{t=3\text{ months}}$$

Naturally, we can combine these equations to directly calculate the weight of the hamsters at 12 months from their weight at birth:

$$\text{Size}_{t=12\text{ months}} = 4.50 * \text{Size}_{t=0\text{ months}}$$

Again, we don't know this is the relationship, so we need to estimate it from the data. All we care about is the size of hamsters at 12 months given their birth size. The simplest way to estimate this relationship is to do a linear regression estimating the final size as a function of the initial size. This regression would result in the following relationship:

$$\text{Size}_{t=12\text{ months}} = 4.65 * \text{Size}_{t=0\text{ months}}$$

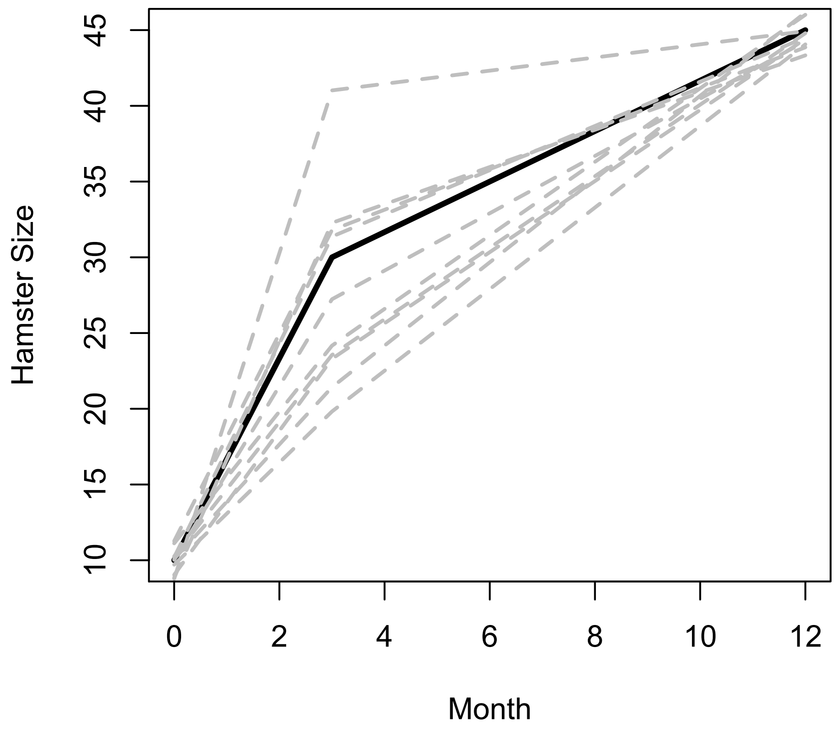


Figure 3. Recorded hamster sizes (dashed grey lines) and the unknown true size trajectory for a hamster starting with size 10 (solid black line).

This result is quite good. The linear coefficient we estimate of 4.65 is very close to the true value of 4.50. Our model so far is doing pretty well.

However, like with the bird migration, someone might point out that this model is too crude. “We know that the hamster go through an infant and juvenile stage”, they might say, “we should model these stages separately so the model is more accurate.”

This viewpoint moreover has actually been held to be the case in the law. For instance, there have been judicial decisions that “life-cycle” models, those that model each stage of an animal’s life are the only valid ones[7]. If we were

[7]Technically the determination is that life-cycle models are the “best available science”. These decisions are misguided and frankly wrong, but that is what occurs when judges are

presenting this model in to an audience that believed that, we would have to create two regressions: one for the infant stage and one for the juvenile stage.

Using the data we have, we would obtain these two regressions:

$$\text{Size}_{t=3\text{ months}} = 2.74 * \text{Size}_{t=0\text{ months}}$$
$$\text{Size}_{t=12\text{ months}} = 1.54 * \text{Size}_{t=3\text{ months}}$$

Combining these regression to get the overall size change for the 12 months we obtain the following:

$$\text{Size}_{t=12\text{ months}} = 4.22 * \text{Size}_{t=0\text{ months}}$$

Now, in this illustrative example we are fortunate to know that the true growth multiplier should be 4.50 so we can test how well our regression actually were. The error for this relatively detailed life-cycle model is $(4.50 - 4.22)/4.50$ or 6.2%. For the "cruder" model where we did not attempt to model the individual stages, the overall error is $(4.50 - 4.65)/4.50$ or 3.3%.

So by trying to be more accurate and detailed, we built a more complex model that has almost twice the error of our simpler model! Let's repeat that: The more complex model is significantly worse in accuracy than the simpler model.

Why is that? We can trace the key issue back to the problem that our data for the 3 month period are significantly worse than our data for 0 months or 12 months. By introducing it into the model, we bring down the overall quality of the model by injecting more error into it. When someone comes to you asking you to add a feature to a model you have to consider if this feature may actually introduce more error into the model as it did in this example.

We can think of life-cycle and many other kinds of models as a chain. Each link of the chain is a sub-model that takes data from the previous link, transforms them and feeds them into the next link. Like a chain, models may only be as good as their weakest link. It is often better to build a small model where all the links are strong, than a more complex model with many weak links.

Overfitting The act of building models that are too complex for the data you have is known as "overfitting" the data[8]. In the model of hamster sizes, the model where we look at each life stage separately is an overfit model; We do not have the data to justify this complex of a model. The simpler model (ignoring the different stages) is superior.

Overfitting is unfortunately too common in model construction. Part of the reason is that the techniques people use to assess the accuracy of a model are

put in the position of making highly technical scientific decisions.

[8]The reverse – building models that are too simple – is called "underfitting". In practice, underfitting will be less of a problem as our natural tendency is to overfit.

often incorrect and inherently biased to cause overfitting. To see this, let's explore a simple example. Say we want to create a model to predict the heights of students in high schools (this is seemingly trivial, but bear with us). To build the model we have data from five hundred students at one high school.

We begin by averaging the heights of all the students in our data set and we find that the average student height is 5 feet 7 inches. That number by itself is a valid model for student height. It is a very simple model[9], but it is a model nonetheless: Simply predict 5 feet 7 inches for the height of any student.

We know we can make this model more accurate. To start, we decide to create a regression for height where gender is a variable. This gives us a new model which predicts women high-school students have a height of 5 feet 5 inches on average, while men have a height of 5 feet 9 inches on average. We calculated the R^2 for the model to be 0.21.

That's not bad, but for prediction purposes we can do better. We decide to include students' race as a predictor as we think that on average there might be differences in heights for different ethnicities. We complete this extended model including ethnic status as a predictor alongside gender and the R^2 fit of our model increases to 0.33.

We still think we can do better though, so we add age as a third predictor: We hypothesize that the older the students are, the taller they will be. The model including age as an additional linear variable is significantly improved with an R^2 of 0.56.

Once we have built this model, we realize that maybe we should not just have a linear relationship with age because as students grow older, their rate of growth will probably slow down. To account for this we decide to also include the square of age in our regression. With this added variable our fit improves to an R^2 of 0.59.

This is going pretty well, we might be on to something. But why stop with the square; what happens if we add higher order polynomial terms based on age? Why not go further and use the cube of age. The fit improves slightly again. We think we are on a roll and so we keep going. We add age taken to the fourth power, and then to the fifth power, and then to the sixth, and so on.

We get a little carried away and end up including 100 different powers of age and each time we add and new power our R^2 gets slightly better. We could keep going, but it's time to do a reality check.

Do really we think that including AGE^{100} made our model any better than when we only had 99 terms based on age? According to the R^2 metric it did (if only by a very small amount). However, we know intuitively it did not. Maybe the first few age variables helped, but once we get past a quadratic

[9] Statisticians would call this the "null" model, the simplest model possible.

$(\mathrm{AGE} + \mathrm{AGE}^2)$ or cubic $(\mathrm{AGE} + \mathrm{AGE}^2 + \mathrm{AGE}^3)$ relationship, we probably are not capturing any more real characteristics of how age affects a person's size.

Variables	R^2
Gender	0.21
Gender, Race	0.33
Gender, Race, Age	0.56
Gender, Race, Age^2	0.59
Gender, Race, Age^2, ..., Age^{100}	0.63
Gender, Race, Age^2, ..., Age^{500}	1.00

So why does our reported model accuracy – R^2 – keep getting better and better as we add these higher order power terms based on age to our regression?

This question is at the heart of overfitting. Let's imagine taking our exploitation of age to its logical conclusion. We could build a model with 500 different terms based on age ($\text{AGE} + \text{AGE}^2 + \text{AGE}^3 + ... + \text{AGE}^{500}$). The result of this regression would go through every single point in our population of five hundred students[10].This model would have a perfect R^2 of one (as it matches each point perfectly) but we know intuitively that it would be a horrible model.

Why is this model so bad? Imagine two students born a day apart one with a height of 6 feet 2 inches the other with a height of 5 feet 5 inches. Our model would indicate that a single day caused a 7-inch difference in height. Even more ridiculous, the model would predict a roller coaster ride for students as they aged. They would gain inches one day (according to the model) and lose them the next. Clearly this model is nonsensical. However, this nonsensical model has a perfect R^2, it is a paradox!

The key to unlocking the solution to the paradox and overcoming overfitting turns out to be surprisingly simple: *assess the accuracy of a model using data that were not used to build the model.*

The reason our overfit model for students looks so good using the R^2 error metric is that we measured the R^2 using the same data that we just used to build the model. This is an issue as we can force an arbitrarily high R^2 simply by continually increasing the complexity of our model. In this context the R^2 we are calculating turns out to be meaningless.

What we need to do is to find new data – new students – to test our model on. That will be a more reliable test of its accuracy. If we first built our model and then took it and applied it to a different high school and calculated the R^2

[10]Remember a polynomial equation with two terms can perfectly pass through two data points, an equation with three terms can perfectly pass through three points, and so on.

using this new data, we would obtain a truer measure of how good our model actually was.

Figure 4 illustrates the effect of overfitting using observation from 9 students. The top three graphs show plots of the heights and ages for these nine students. We fit three models to these data: a simple linear one, a quadratic polynomial, and an equation with nine terms so that it goes through each point exactly.

Below the three graphs we show the regular R^2 that most people use when fitting models, and also what the true R^2[11] would be if we applied the resulting model to new data. The regular R^2 always increases so if we used this naive metric we would always end up choosing the most complex model. As we can see, the true accuracy of the model decreases after we reach a certain complexity. Therefore the middle model is really the better model in this case. When illustrated like this, this concept of overfitting should make a lot of sense; but, surprisingly, it is often overlooked in practice even by modeling experts.

In general, overfitting should be watched for carefully. If you do not have a good metric of model error, the inclination to add complexity to your model will be validated by misleadingly optimistic measures of error that make you think your model is getting better when it is actually getting worse. The optimization techniques we described earlier in this chapter are also susceptible to these problems as every time you add a new variable to be optimized the optimization error will always go down further (assuming the true optimal parameter configuration can be found). The more parameters you add the worse this effect will be.

How do we estimate the true error of the model fit? The simplest approach is to take your dataset and split it into two parts. Build the model with one half of the data and then measure the accuracy using the other half. So with our high-school students we would randomly assign each one to be used either to build the model or to assess the model's error. Advanced statistical techniques such as *cross-validation* or *bootstrapping* are other approaches and can be more effective given a finite amount of data. Unfortunately, we do not have space to discuss them here, but we would recommend the reader explore them on their own if they are interested in this topic.

No one ever got fired for saying, "Let's make this model more complex." After this chapter, we hope you understand why this advice, though safe to say, is often exactly the wrong advice.

[11] You might have heard of R^2 variants such as the Adjusted R^2. The Adjusted R^2 is better than the regular R^2; however it is important to note that it is not the true R^2. Adjusted R^2 also has some issues with overfitting.

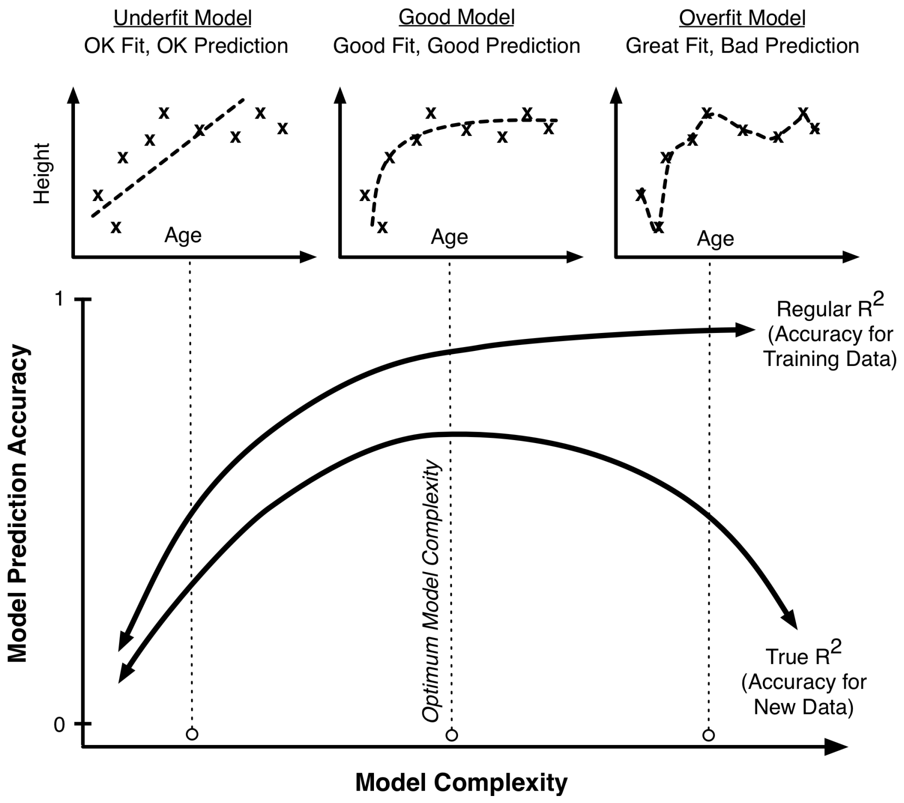

Underfit Model
OK Fit, OK Prediction
Good Model
Good Fit, Good Prediction
Overfit Model
Great Fit, Bad Prediction
Height
Age
Age
Age
1
0
Model Prediction Accuracy
Regular R^2 (Accuracy for Training Data)
True R^2 (Accuracy for New Data)
Optimum Model Complexity
Model Complexity

Figure 4. Illustration of overfitting. The best model is not necessarily the one that fits the data the closest.