[DRAFT] Beyond Connecting the Dots: Mastering the Hidden Connections in Everything that Matters

Gene Bellinger Scott Fortmann-Roe

2013-06-22

Contents

Chapter 1

Preface

Ludwig von Bertalanffy(1) first proposed, in 1937, that the same basic structures operated across all disciplines, and if one learned how these structures operated one could transfer much of their learning from one discipline to another. When moving from one discipline to another, one would simply have to learn the structures that were operating, and the labels on the elements of the structures. On first reading this may seem most profound, or maybe even preposterous.

However, if you think about it, maybe there is some truth to it after all. What follows is the introduction to a live Systems Thinking book presented from a cross discipline models perspective. Live in the sense that the models are presented in a form that allows you to actually interact with them.

von Bertalanffy wrote "Allegemein Systemlehre" which was translated into English as "General Systems Theory"(2) and I expect we've still not recovered from the translation error. What he intended was a "General Theory of Systems" or "General Systems Teaching," a way to support learning about the structures which operated across all disciplines. Today there are a set of structures referred to as Systems Archetypes which I believe are just what Bertalanffy had in mind.

In the words of von Bertalanffy, "The student in 'system science' receives a technical training which makes systems theory – originally intended to overcome current overspecialization – into another of the hundreds of academic specialties"(1)

Systems Thinking is not a method though more of a way of looking at the world around us and understanding based not from understanding things though more from understanding relations and interactions between things. And while there are many who believe that Systems Thinking or a Systems Perspective provides the best foundation for creating effective approaches of dealing with challenges and shaping a better tomorrow. Yet even with that view, over the past 75 years it has not become widely adopted, even though during that period dozens of approaches have been developed with claim to embrace the Systems Thinking world view. I believe Pogo had it right when he said, "We have met

the enemy and he is us." I have repeatedly commented to people that the greatest impediment to the adoption of Systems Thinking is Systems Thinkers.

This should provide you with a sense of why this book has to be different. Now let me offer you a view of how it will be different.

It is our intent to provide a basis for recovering from this overspecialization by offering an extensive series of models from everyday life that will show the value of looking at things though a different lense. We will then build on this to develop an understanding without all the terminology and complexity that typically drives people away from Systems Thinking.

References

- Davidson, Mark. 1983. Uncommon Sense: The Life and Thought of Ludwig von Bertalanffy http://www.amazon.com/Uncommon-Sense-Thought-Bertalanffy-1901-1972/dp/087477165X/

Chapter 2

Chapter 1 - It's The Pattern That Connects - v2 13.04.08

Notes to Self!

remove it and it's and I most everywhere.

If you give people the answers there is no need for them to practice insight and the purpose of BCTD is to encourage the practice of gaining insight into why systems behave the why they do.

Notes to Reviewers

v2 Modifications

- Ladder of Inference section removed so there is no Figure 4 at the moment.
- The embedded Insight Maker are more interactive because of an enhancement Scott has made to Insight Maker. Please run the models in the web page.
- The comparison structures model of Figure 23 has been completely overhauled.

Chapter Intent

Develop an awareness that the diverse world around us has a commonality that can be meaningfully represented by just a few interacting elements with rather simple attributes. The basic operation and interaction with embedded models must also be experienced and supporting aspects of Insight Maker explained.

Figure Captions

Each figure is followed by a sequenced figure caption line that starts with ** and these lines are also an internet link. These lines are inserted so I can easily get back to wherever that graphic originated should I need to create a revised version of it. These statements will be deleted by the post processor and replaced with figure captions which are embedded in the Markdown formatting.

Insight Maker References

I'm doing the best I can representing the version of Insight Maker I won't be able to see for a couple of months. The interactive Insight Maker models are embedded from Insight Maker and the model is owned my me. This means that when one looks at it in this chapter it doesn't look like it will look in the final book. Scott is creating a version of Insight Maker that will operate in a touch tablet environment. That version of Insight Maker will be embedded in the book and each book owner will own the models in the book. That means they will look different. As such I have to code something so I'll know what to include later and reviewers can look at and connect with the written words associated with it. Getting through this seems to be a tall order.

Macros

There are certain aspects of the text formatting we don't have figured out and have resigned ourselves to the fact that we won't have this figured out for some time. As such macros are being coded to be replaced in the content post processing phase. I sorry that it's likely to make the text a bit more difficult to read.

- **model attribute**
- ç
- `equation`
- [**model primitive**]
- **ui reference**

Relation to Table of Contents

What follows was presented in the Table of Contents as three separate chapters though the writing seemed to get away from me. The may be split into several chapters or the table of contents may be corrected. Presently it's a bit difficult to tell.

What you learn, and your capacity to learn, serves as the basis for everything you do in your life. Yet, have you ever thought about how you really learn

about the world around you? Yes, there are some things you memorize early in life, like the times tables, and you learn to remember these, though is that really learning? Do you remember that if you put your hand on something very hot it will burn you, or is that something you learned? And if you learned that, how was it that that learning happened?

Consider the following

- I have a box that's about 3' wide, 3' deep and 6' high
- It's a rather heavy box
- The has a couple of doors on it
- When you open the doors it's cooler inside the box than outside
- One compartment is much colder than the other
- When you open the door a light comes on
- There's food inside the box
- The box is in the kitchen
- There are sticky notes all over the front of the box
- There's a collection of papers and stuff on top of the box
- If you move the box you'll probably find a lot of dust under it
- The box is plugged into an electrical outlet
- From time to time you can hear the box running

At some point in this sequence you probably became convinced that what was being described was a refrigerator. Now stop for a moment and ask yourself just how was it that you realized what was being described was a refrigerator? Yes it would have been easier if I had just shown you a picture of a refrigerator, though that would have spoiled it, wouldn't it.

As long as you knew beforehand what a refrigerator was, the statements could have been given to you in any order, and still at some point you would have finally realized what was being described. If you had never seen, nor heard about, a refrigerator before you would still be wondering what was being described and what to call it.

You have also most likely come to understand that all refrigerators are not identical. Some have one door with a separate compartment inside. Some have two doors and a drawer. Some are much smaller than others. Some can fit under a counter and some even fit on top of a counter. Some can be so large you can walk into them.

If you see any of these you quickly decide it's a refrigerator. How does that happen? Gregory Bateson, one of the great thinkers of our time, said, "It's the pattern that connects." If you reflect on this statement you should come to realize there are actually different ways to interpret what it means. In this particular case the pattern connects you to the following purpose

Figure 1. From the description you knew it was a refrigerator - but how?

Figure 2. Many kinds of refrigerators, or freezers - But how do you know?

- The box keeps food from readily spoiling by keeping it cold
- Part of the box is a freezer which keeps food from spoiling for even longer

and you understand it to be a refrigerator. Though now that we've arrived at this point we still haven't addressed the question of how you know. You probably were not actually taught that it's the above purpose that defines the essence of a refrigerator. Most people were not, though they have essentially learned it over time.

Models

Models are the way we look at, and understand the world around us. All we have are our models. They are the way we understand everything. This is so because we build our understanding based on what we already understand. The world around us simply has too much detail for us to pay attention to everything. A refrigerator has many pieces though how many do you really pay attention to? Probably not many unless you build or repair refrigerators. We filter out much of the detail around us so we don't become overloaded and we choose what to pay attention to. Sometimes we do this consciously and sometimes subconsciously. In the midst of what we choose to pay attention to there are patterns. Whether we realize it or not it is these patterns that we pay attention to and attempt to make sense of. We understand these patterns by linking them to extend patterns we already understand. And much of the world around us we simply ignore for if we didn't we would just become overwhelmed.

Remember

A model is a simplified version of some aspect of the world around us to help us understand something.

Learning

When we experience something that experience falls somewhere between complete novelty, meaning that we can't connect it with anything in our past experience, and complete confirmation, meaning that it represents something we perceive as already completely understood. The things we experience which lie somewhere between complete novelty and complete confirmation provide a basis for learning. They represent a basis for connecting to understood patterns, extending our understanding, and what results is learning. {Cite: Jantach, Eric. 1980. The Self-Organizing Universe: Scientific and Human Implications. Pergamon Press. http://www.amazon.com/The-Self-Organizing-Universe-Implications-Innovations/dp/0080243118/}

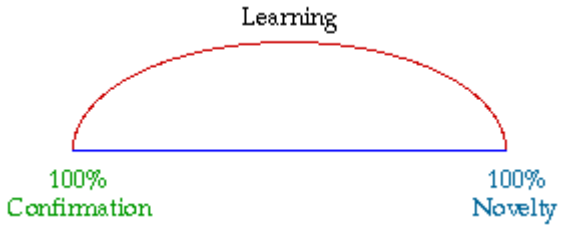


Figure 3. Experience between novelty and confirmation as a basis for learning

Consider running into a refrigerator that looks like no refrigerator you've never seen before. From an initial view you are likely not to perceive it as a refrigerator. As you inspect it to find it serves the purpose you've come to understand for refrigerators or if someone tells you it's a refrigerator you then expand or extend your awareness of the range of patterns that constitute a refrigerator.

A Basis for Flawed Learning

While reading the previous paragraphs did it dawn on you that much of this pattern recognition/connection/extension learning doesn't happen consciously? We connect with patterns and extend our knowledge at times without even being consciously aware that it is happening. And when it happens in an subconscious manner there isn't really any critical validation that happens

along with the learning. Because this ongoing learning happens without critical validation there are things we learn and come to believe which are actually incorrect. We have perceived patterns and extended our learning in a flawed manner. The really annoying thing is that we then act on these beliefs, and when we produce results that don't go the way we planned we wonder why. Or even worse, we don't actually learn from the results and correct our flawed models.

When we act on flawed beliefs when attempting to solve problems we typically create more problems than we fix. It has been said repeatedly that the majority of today's problems are the direct result of yesterday's solutions. Wouldn't this provide a sense that we might really benefit from a better way to think about the world around us, develop better understanding, and develop solutions that don't come back to haunt us in the future?

A Better Way

Based on the understanding I hope you've developed to this point it should be obvious that we could benefit from a better way to develop models of what we believe that are more likely to be correct as well as surface flaws in many of our current beliefs.

Ludwig von Bertalanffy first proposed, in 1937, that the same basic structures operated across all disciplines, and if one learned how these structures operated one could transfer much of their learning from one discipline to another.{Davidson, Mark. 1983. Uncommon Sense: The Life and Thought of Ludwig von Bertalanffy. J.P. Tarcher, Inc. http://www.amazon.com/Uncommon-Sense-Thought-Bertalanffy-1901-1972/dp/087477165X/} When moving from one discipline to another, one would simply have to learn the structures that were operating, and the labels on the elements of the structures. On first reading this may seem most profound, or maybe even preposterous. However, if you think about it, maybe there is some truth to it after all.

I'm not asking you to believe the previous statement just because it was provided here. Though if you give me a few minutes the experience that follows may allow you to arrive at a sensibility of the statement from your own perspective.

Consider the images in Figure 5 and ask yourself what it is that all these different items actually have in common.

Each of these items represents a collection of stuff. Admittedly each image represents different stuff though stuff just the same. Because in each case this stuff collected over time it's really more appropriate to refer to the the collections as accumulations. And as you will come to realize it is extremely important to remember that accumulations take time to accumulate, and often even longer to get rid of when you find out you don't want them.

The shorter term often used to refer to an accumulation is "stock." Just where this term originate I'm unsure and what you call an accumulation of stuff isn't

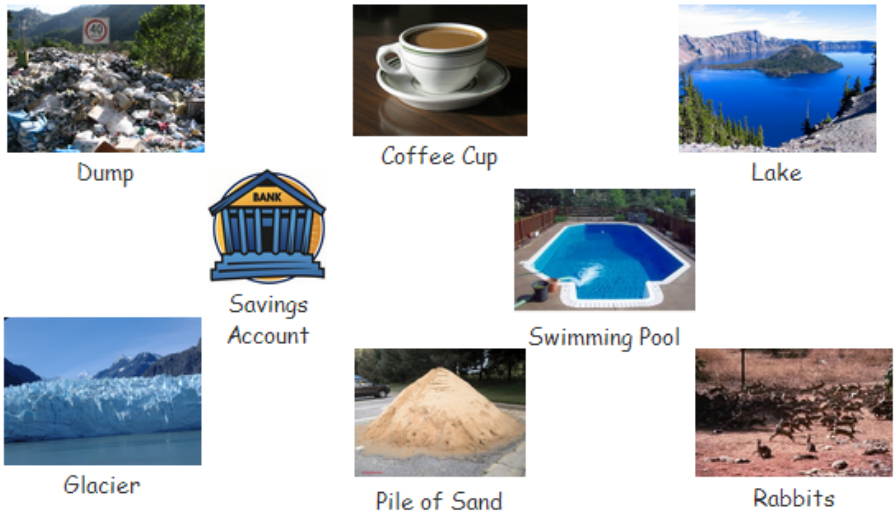


Figure 5. What do these items have in common?

nearly as important as remembering it's a bunch of stuff that collected over time. How much time is different for each one of the accumulations and now it's probably time to talk about how accumulations happen over time.

For each of the accumulations in Figure 5, how they change is a bit different, as are the time frames concerned. Time frame being the time it takes for some real noticeable change in the accumulation. Let me describe each in some detail.

Coffee Cup

You usually fill a coffee cup from a coffee pot and it takes a few seconds. Then you take a few minutes to drink the coffee as it's usually too hot to drink when you initially get it.

Dump

Generally a dump accumulates by the truckload after the garbage is picked up at houses or businesses in your community. If the dump were just getting started you'd probably notice it grow with each additional truck load. As it gets bigger and bigger it's gets more difficult to notice that it's growing, even though it is. While the dump is likely to grow almost every day we are probably more likely to think about the growth of the dump in months and years. And does it ever really go away? Usually when it gets to be too much a new dump is started somewhere else and the current dump is buried. Though when it's buried it doesn't really go away. It's still there and we'll probably talk more about dumps later on.

Glacier

A glacier is a long term accumulation of snow which packs down and turns to ice. Glaciers get bigger in the winter when snow falls and they get smaller in summer when some portion of the glacier melts. The time frame one usually uses to think about glaciers is years or even decades.

Lake

Lakes are bigger than a pond and smaller than an ocean and usually filled with fresh water, not salty that is. The lake is filled by rivers and streams that flow into it as well as rain water. One might think of this in terms of gallons per hour or gallons per minute in the case of a large inflow such as at Niagara Falls where the water flows into Lake Ontario in the USA. Water leaves the lake through rivers and streams as well as evaporation into the air. For a lake one might think about the water flowing into our out of the lake in hours though when considering the level of the lake itself the change might be considered over days or weeks. It sort of depends on what you're interested in.

Pile of Sand

The pile of sand probably showed up in a truck that dumped it right where it is. While it may have taken the truck a while to drive from the wherever it started it probably only took a couple of minutes to dump the truck once it arrived. And the sand is probably referred to in cubic yards, which is how much sand it takes to fill a box that's 1 yard wide, 1 yard deep, and 1 yard high. How long it takes for the sand to go away depends on how it's taken away. If you use a wheel barrow then you have to shovel the sand into the wheel barrow and take it to wherever you're going to use it. At this rate it may take days to move it. If you move it with a small piece of machinery, a Bobcat or a Backhoe, then will will probably only take a few minutes to an hour to get it moved.

Rabbits

A population of rabbits gets larger with new rabbit births and gets smaller with rabbit deaths. Have you ever heard the phrase "multiply like rabbits?" What it means is that it doesn't take very long for a few rabbits to become many rabbits, as long as there is a good food supply and not to many predators like wolves and coyotes. The time frame for considering a rabbit population is probably months to years.

Savings Account

A savings account is a bank account where if you put money and if you keep it there the bank will periodically give you money just for keeping it there. They won't give you very much, though some. If you keep putting money in your savings account every so often and never take it out one day you'll be rich. Yet,

for some reason that doesn't happen to too many people. We'll have to talk about that sometime later in the book. One generally thinks about the money associated with a savings account in dollars, the interest rate as a percentage, and the time frame in months and years.

Swimming Pool

Swimming pools usually hold thousands of gallons of water and you usually have a couple of options to fill one. You might use a garden hose, which will take days, or a hose from a fire hydrant, which will take a few hours, or from a tanker truck, which probably takes a few loads. In each case the water filling the pool is probably measured in gallons per hour. Once you fill the pool you loose a little water when people get in and out of the pool, thought not too much. Most of the water loss from a pool is though evaporation due to the sun and when you backwash the filter used to keep the pool clean. The change in amount of water is usually measured in gallons per hour.

Exercise

Take a few minutes and identify half a dozen situations you're familiar with where there are stocks that accumulate over time. What are the quantities for those stocks, e.g., gallons, pounds, kilograms, etc.. What are the flows that increase and decrease them and what are the time frames over which you think about the accumulation of that stock?

At this point you may be wondering why so much time was spent making you walk though all these examples for the accumulation of stuff. Since we said this was an interactive book you're probably wondering where the interaction is.

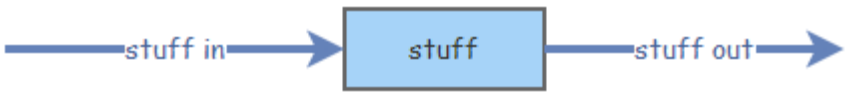


Figure 6. The Accumulation of Stuff

All the accumulations depicted in Figure 5 can be represented in a general form by the model in Figure 6. Remember we defined a model as a simplified version of some aspect of the world around us to help us understand something. It doesn't get much simpler than this does it?

Some amount of stuff flowing in causes stuff to increase over time and stuff flowing out causes stuff to decrease over time. With both of these happening at the same time stuff increases if stuff in is larger than stuff out. And if stuff out is greater than stuff in then the accumulation of stuff gets smaller. The most critical aspect of this to remember is that it takes time for stuff to increase or decrease. How fast the change happens depends on the amount of stuff in the flows.

Lets take a specific instance. Figure 7 represents Figure 6 in Insight Maker, an interactive modeling environment. We'll talk about how this was done shortly. Now suppose we have a swimming pool and we start filling it with a hose that fills at 50 gallons an hour. If we let the hose run for 24 hours how much water will be in the pool? Admittedly the math is pretty straight forward though the idea here is to show how you can use a model to show changes over time.

If I set up the model in Figure 7 with stuff = 0, stuff in = 50 and stuff out = 0, set the Time Settings for 24 hours, and then click the Run button, the model produces the graph in Figure 8.

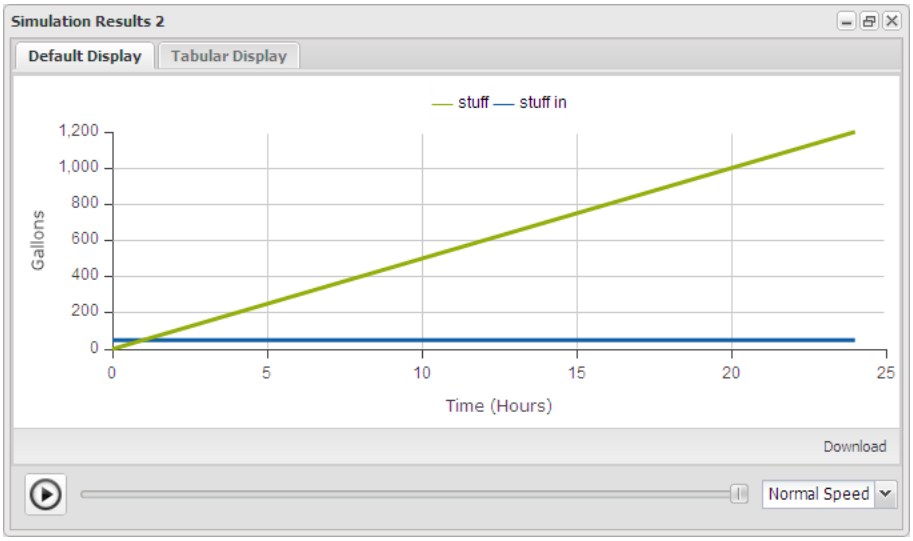


Figure 8. Adding water to the swimming pool

This graph indicates that the after 24 hours the swimming pool with have 1,200 gallons of water in it. I know, it's about as interesting as watching paint dry. Actually, as you will come to find out, that's a good thing because this is really easy. A more interesting question might be, if the swimming pool holds 20,000 gallons of water how long with it take to fill with water at 50 gallons per hour? We'll get to this shortly.

Modeling Notes

As various models are developed and used I'll present aspects of the modeling environment that you'll need to do the exercises. I won't spend a lot of time on pieces you're not going to use immediately so please don't let any of the displays overwhelm you. I think it's far easier to remember things when you actually use them.

Canvas

The center area is the work area where you create models. This area may be scrolled if necessary. I'll talk about how to actually create models in the next chapter.

Stock

A rectangle indicates a quantity of something that accumulates, and accumulation takes time. Stocks don't change in the blink of an eye, well unless you blink for a long time.

Flow

A directed arrow representing the flow of something into or out of a stock. Remember that a stock can only be changed by a flow. Hand waving and magic don't work. The flow has to be explicit to cause a stock to change, and it takes time.

Toolbar

include another graphic to make the upper right arrow clear

Notice in the upper right corner there is a small down arrow. If you click on this arrow it will open the toolbar displayed in Figure 9. The toolbar contains all the tools you will use to build and modify models. Yes, you get to do everything on a single screen, with a few pop up windows of course.

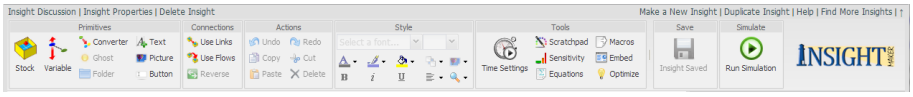


Figure 9. Toolbar

Parameter Tab

Just below the arrow you clicked to open the top toolbar is a right pointing double caret. If you click this the parameter tab will close and the right pointing

double caret will now point left and can be used to open the parameter tab. This tab serves two different purposes.

If there are no elements of the model selected on the canvas the parameter tab will be similar to Figure 10 and contain the model description, tags, and parameter sliders used to set parameter values just before running the model.

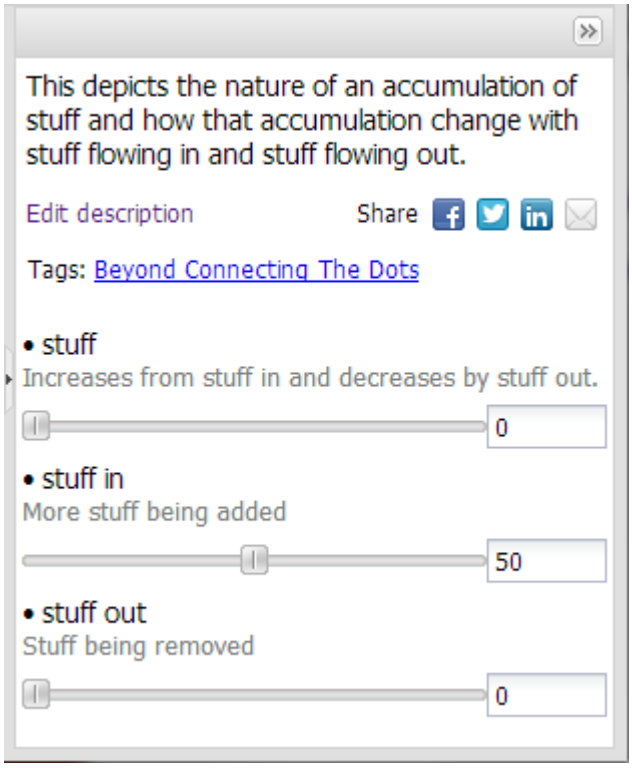


Figure 10. Configuration Panel

If there is a single element selected on the canvas then the parameter tab will present the list of parameters that can be set for that element. Figure 11 shows the parameters for the stuff element of the model. This is where I set the stuff in to 50 before running the model. Please don't be overwhelmed by this long list of parameters. We'll cover them one at a time as they are actually used in a model.

You should note in Figure 11 under the User Interface section it indicates that there should be a slider for stuff and it can be set for values from 0 to 100. Each element has some of the same parameters and some unique to it. Click one of the flows and see what its parameters are.

Just a couple more pieces and you can go interact with the model some more and get away from this boring description.

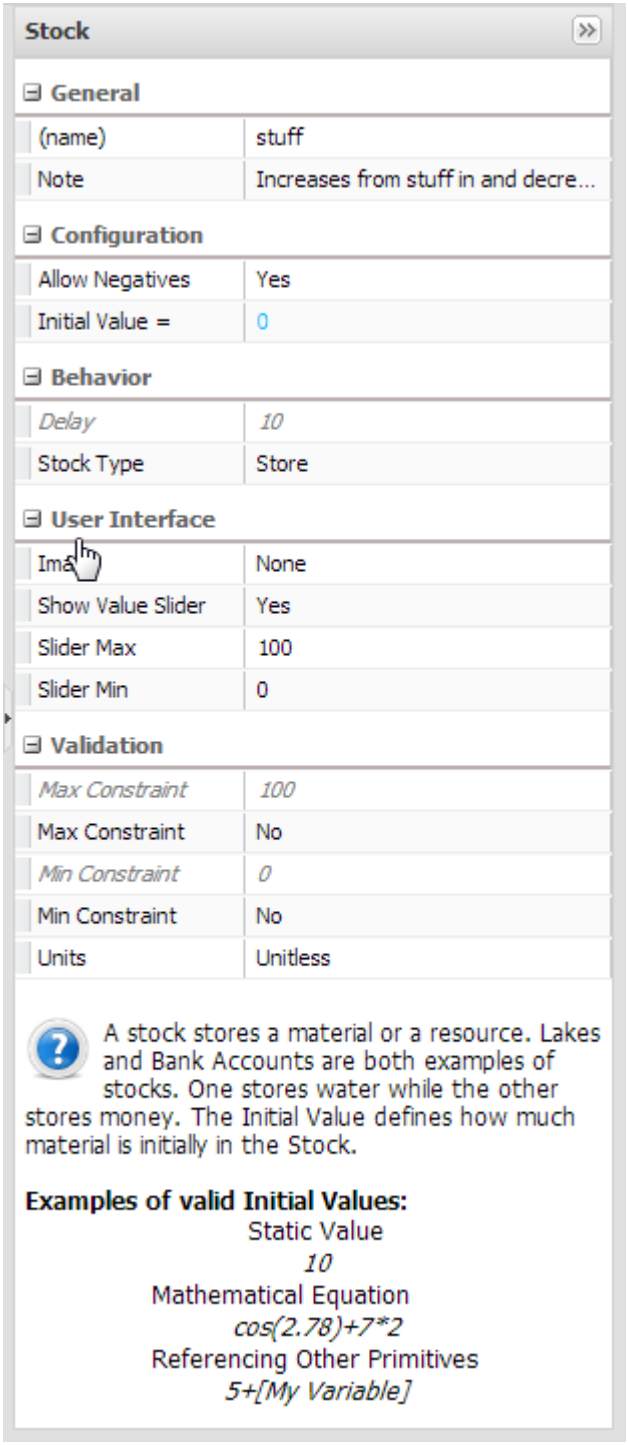


Figure 11. Element Parameters

Time Settings

In Figure 8 I talked about the swimming pool filling for 24 hours. It's the Time Settings tool that allows you to define this for the model. Figure 12 shows the elements you can set before running a model.

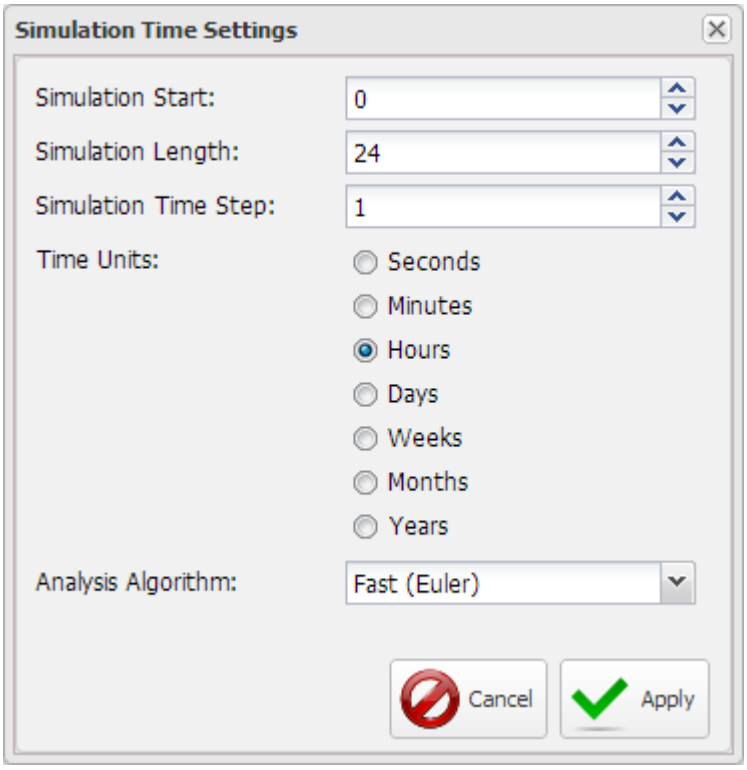


Figure 12. Time Settings

This is where I told the model to start at Time = 0 and runs for 24 time units. It steps one unit at a time and the unit is in Hours. Don't worry about Time Step for now. We'll get into that later.

Simulation Results

When you click the Run button the model is stepped through the defined time period and produces a display of the results. There are various options for the type of display and which elements are displayed as in Figure 13.

Configure Simulation Results

A default configuration is put together when the model is constructed on the canvas. If you click the Configure button in the upper right corner of the

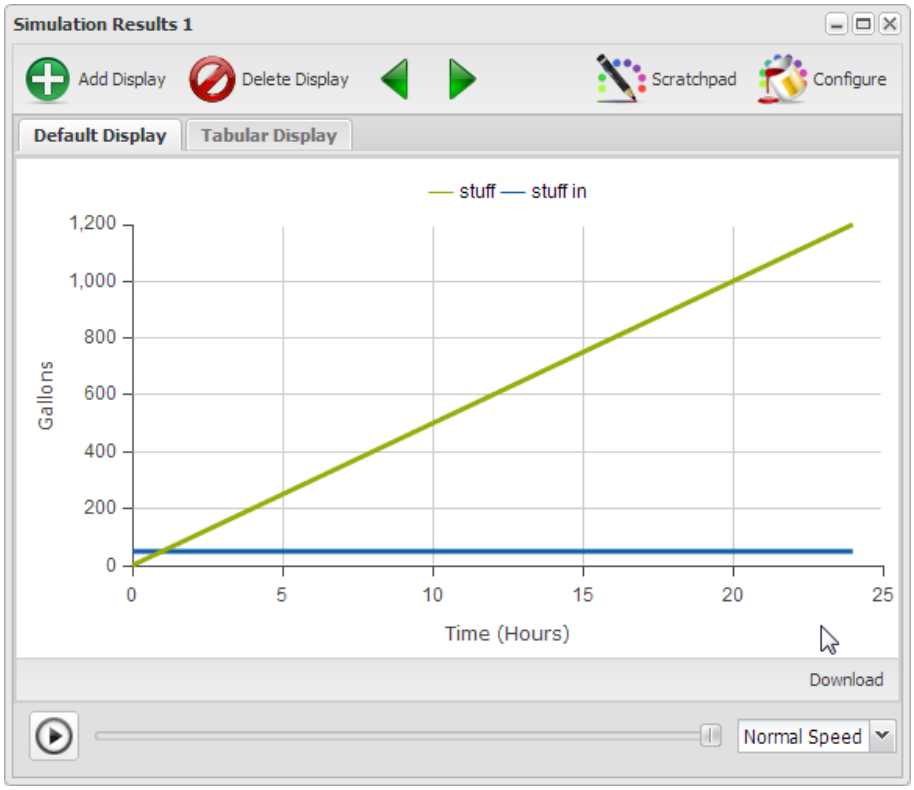


Figure 13. Simulation Results

Simulation Results window the Chart/Table Configuration window will open. It is in this window you indicate what type of display you want and which items of the model are to be displayed. The only part you need to be concerned about at the moment is the Y-Axis Label field. That's where I indicated that the items displayed were in Gallons. You will need to change this shortly in the next exercise.

Note that if you change items in the configuration they will be immediately reflected in the Simulation Results window when you click Apply. You don't need to run the model over again to see a different configuration of the data. This makes it very convenient when when you decide you need another display for one or two of the items.

I hope you haven't found this short introduction to the modeling environment too overwhelming. As I said I will try to introduce different parts of the environment just as you need them to interact with the models presented.

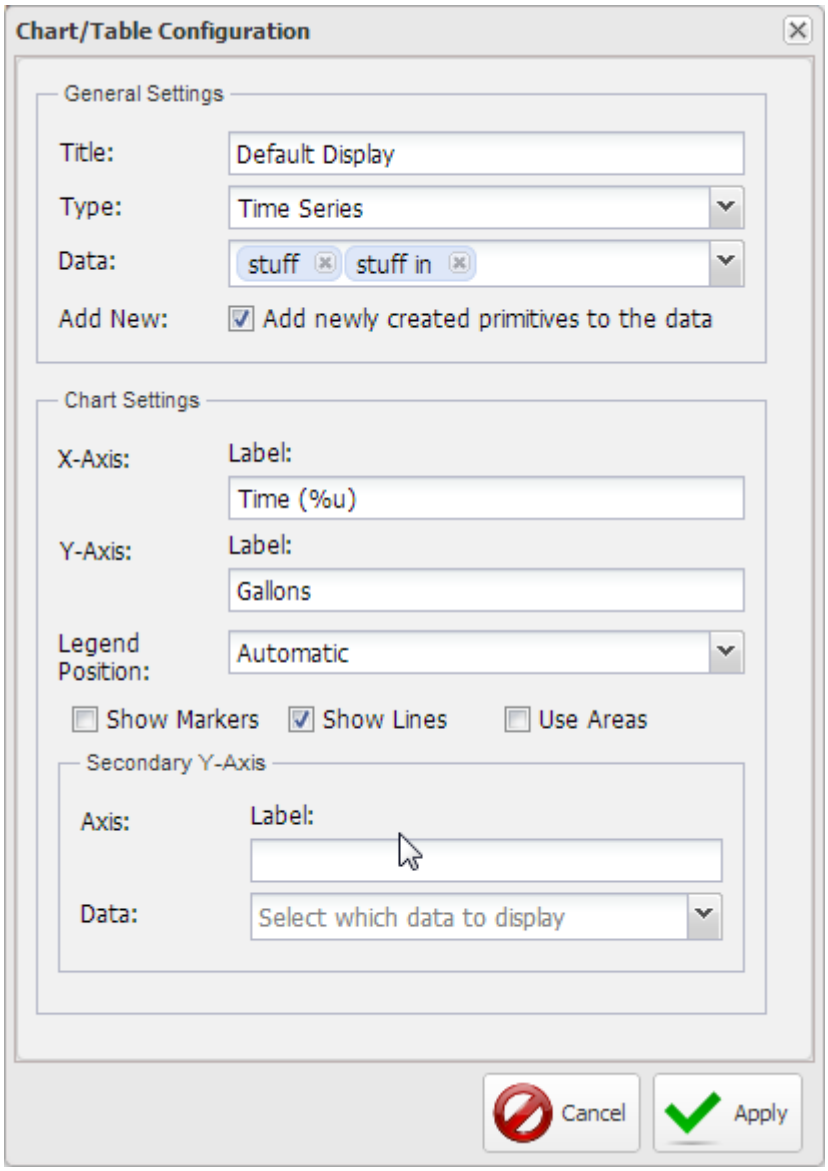


Figure 13. Chart/Table Configuration

Too much explaining and not enough hands on interaction gets to be real boring in a hurry. I encourage you to actually do the exercises presented. By interacting with the various aspects of the modeling environment you will develop a level of comfort and expertise which will serve you well throughout the rest of the book.

Exercise

Go back and consider the various pictures in Figure 5. Pick a couple of them to model. The only parts you need to set up are the Time Settings, how long will it run and the Time Units. You can also set the values for stuff, stuff in and stuff out on the Configuration Panel. After you run the model open the Chart/Table configuration window and set the Y-Axis Label appropriate for what you're modeling. I encourage you to be adventurous. Make new Displays, Table Displays, etc. You can't break anything, it's just an opportunity to become comfortable with the environment and learn.

Now that you've become intimately familiar with almost the simplest model possible lets go back and look at a couple of the pictures in Figure 5 and think about how the accumulations change in a bit more detail.

Rabbit Population Growth

If you modeled the accumulation of rabbits you may have already realized that the model of Figure 7 is missing something. Yes, if you add rabbits to rabbits you get even more rabbits. Though if you have more rabbits don't they create even more rabbits? Figure 14 is a model that reflects the the notion that rabbits create more rabbits.

Modeling Notes

There are a couple new pieces added into the model here and it's probably a good idea to explain the pieces before talking about how it works. The previous model had a stock, something that accumulates, and flows, the movement of stuff into or out of a stock. And the real important thing to remember is that accumulations take time to change. Stocks only change in the blink of an eye if you blink for a very long time.

Variable

A constant or equation used to influence some part of the model. Remember that a variable and a stock are different. A stock is an accumulation that changes over time as a result of one or more flows. A variable may change though it doesn't represent an accumulation. Rabbit Birth Rate is a variable, and in this model a constant value.

Link

A link is used to communicate a value of one element to another. The link doesn't actually represent something moving like a flow does.

= & i

If you mouse over the elements of the model you'll notice an = and an i appear. The i indicates there is additional info available. If you click the i a note window will open with a description of the element. This info was entered when the model was created. The = indicates there is a value or equation associated with the element. If you click the = it will open the **Equation Editor** window. We'll talk more about this when you start building a model.

Based on the previous modeling notes the model depicted in Figure 14 indicates that if you start with some population of Rabbits and each time period the current number of Rabbits times the Rabbit Birth Rate will result in a number of Births. This number of Births will then be added to the accumulation of Rabbis and figure into the calculation for the next period. If you mouse over the elements of the model and click on the = sign you can look at the definitions for the elements.

The Time Settings for the model were set up to run from 0 to 12 months. If you click the Run button you you might be surprised when the model produces the graph in image in Figure 15.

> The values in figure 14 are supposed to be 0 unless someone changed them.

Figure 15 really shouldn't be a surprise. If you look at the Configuration Panel you'll see that it indicates 0 Rabbits and 0 Rabbit Birth Rate. If there are no Rabbits how could anything happen? And if we had some Rabbits with the Rabbit Birth Rate was 0 what would you expect the result to be?

Suppose we start with 10 Rabbits, half of which are male and half of which are female. My research indicates that a female rabbit can give birth to between 18

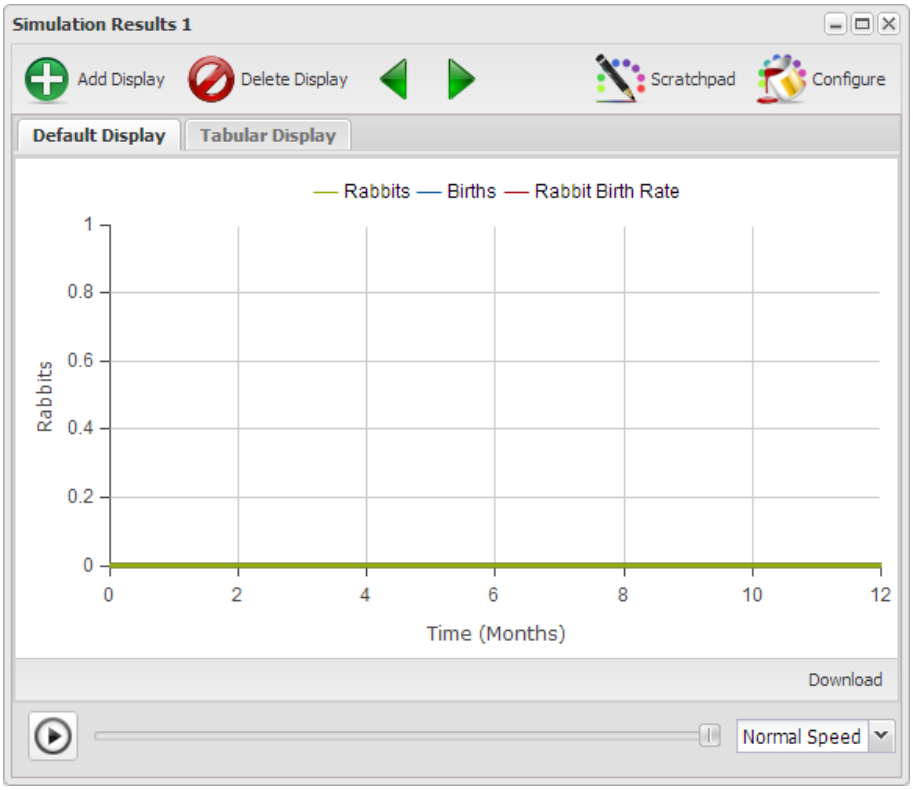


Figure 15. Rabbit Population Growth with No Rabbits

and 26 Rabbits a year. I'll average this out (18 + 24) / 2 = 22 and then I'll round this up to 24 just because it will make the math easier. If a female Rabbit can produce 24 Rabbits in a year, that's 2 per months, though it actually takes two Rabbits. With all these assumptions we get about 1 new Rabbit per month for each Rabbit. If you plug Rabbits = 10 and Rabbit Birth Rate = 1 into the model and run it you should get Figure 16.

Forty thousand Rabbits in a year? That seems a bit bizarre doesn't it? This result actually points out the real value of modeling, which is learning. You build a model based on what you think you understand. You then populate it with assumptions about the values and you run it. The result then either seems to make sense or seems really bizarre. In that case what the model is telling you is that either the structure is wrong, the assumptions are wrong, or both, because the world can't possibly be this bizarre. As a result you investigate the model and your assumptions and as you understand better the model gets better. At some point the model finally serves its purpose, to be a simplification of some aspect of the world which leads to a better understanding. I hope you come to find, as I have, that going round and round with a model can be a

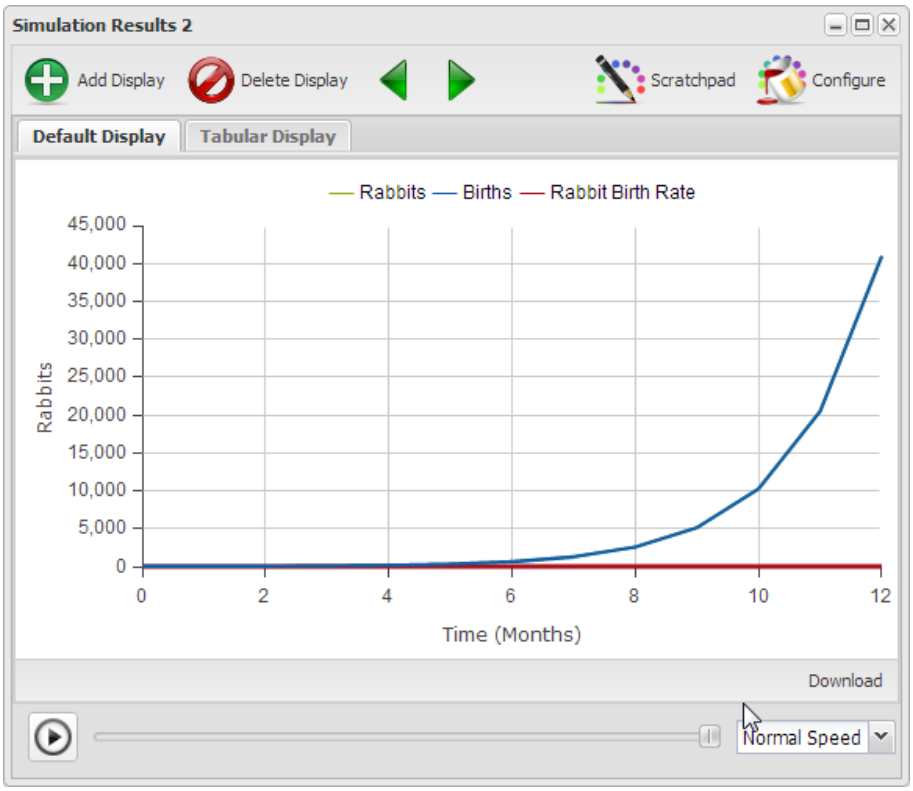


Figure 16. Rabbit Population Growth with 10 Rabbits

delightful learning process.

> note the choppy nature of the graph. talk about why this is the result of the step size and we'll address this in chapter 2

After that sidetrack lets get back to our 40,000 Rabbits that can't possibly exist after a year. I'm pretty sure I can be certain how many Rabbits I started with at the beginning. And when I check my formula for Births = Rabbits * Rabbit Birth Rate it seems to be in order. This sort of means my assumption for Rabbit Birth Rate must be too big. And if you think about what the model is doing it's probably not too difficult to figure out that the model assumes that a Rabbit can be born this month and then give birth to another Rabbit next month. If a Rabbit has to mature for six months before it gives birth to Rabbits then the Rabbit Birth Rate might be something more like 20%. Using this estimate for Rabbit Birth Rate the model produces Figure 17.

Is this right? A good thing to remember at this point is that's actually the wrong question. A better question might be, "What have I learned, and is there

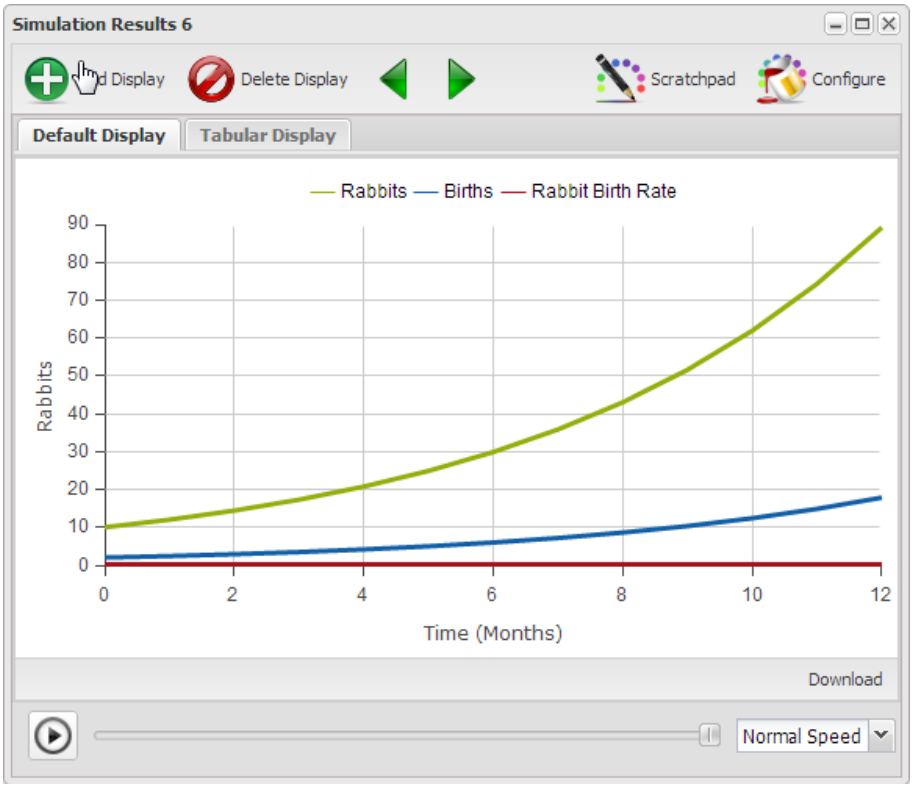


Figure 17. Rabbit Population Growth with 10 Rabbits and 20% Birth Rate

more I can learn?" The graph in Figure 17 sure seems more reasonable than what the model presented in Figure 16 though I don't think we have a high degree of confidence in the current Rabbit Birth Rate. And there are a number of other questions we could ask about our Rabbits. What is the Rabbit Death Rate? Do they have enough food to eat? Are they living out in the open where Coyotes and Foxes can get at them? Does their owner have a passion for Rabbit Stew? These might each be a basis for building a better model, though at this point we're going to leave the Rabbits alone and move on to something else.

The most important learning I hope you take away from this model is that when what flows into the accumulation increases as the accumulation increases the accumulation can get real big in a hurry. This is actually called exponential grown and we'll talk in more detail about this in due quite soon.

Filling A Swimming Pool

Long long ago, meaning back in Figure 7 and Figure 8 I was talking about filling a swimming pool with a hose and how much water was in the pool after

a period of time. A more useful question might be, If the pool holds 20,000 gallons of water and the hose fills the pool at 50 gallons per hour, how long will it take to fill the pool. I know, you can do the math faster than it will take to build the model. Please bear with me a bit as there's another aspect of models right around the corner you will find very useful on an ongoing basis.

I begin with a Swimming Pool that needs to be filled with a hose. I know how many gallows of water it takes to fill the pool and I don't want to put too much water in the pool. I create a model where I compare the amount of the water in the Swimming Pool with the the Full Level and use that to decide whether water is flowing in the hose or not. If you mouse over Hose and click the = sign you'll see the following equation.

> IfThenElse([Swimmng Pool] < [Full Level], [Full Level]-[Swimmng Pool]), 0)

This says that if the Swimming Pool isn't full then I need to add enough water to fill the pool. And if the Swimming Pool is full then I add 0.

Modeling Note

Isn't it curious that the structure of this model looks just like the one for the Rabbit Population growth in Figure 14. We'll come back to this after we figure out how long it's going to take to fill the Swimming Pool.

With the Time Settings set for the model to run for 24 hours. Set the Swimming Pool to 0, meaning empty, and the Full Level to 20,000, on the Configuration go ahead and click the Run button. You should end up with the graph as shown in Figure 19.

> note the choppy nature of the graph. talk about why this is the result of the step size and we'll address this in chapter 2

This is really great. We can fill the Swimming Pool in just 1 day, or can we? Either it's a really really big hose or we've done something wrong because it's probably not really possible fill the Swimming Pool with a Hose in one day if it takes 20,000 gallons or water.

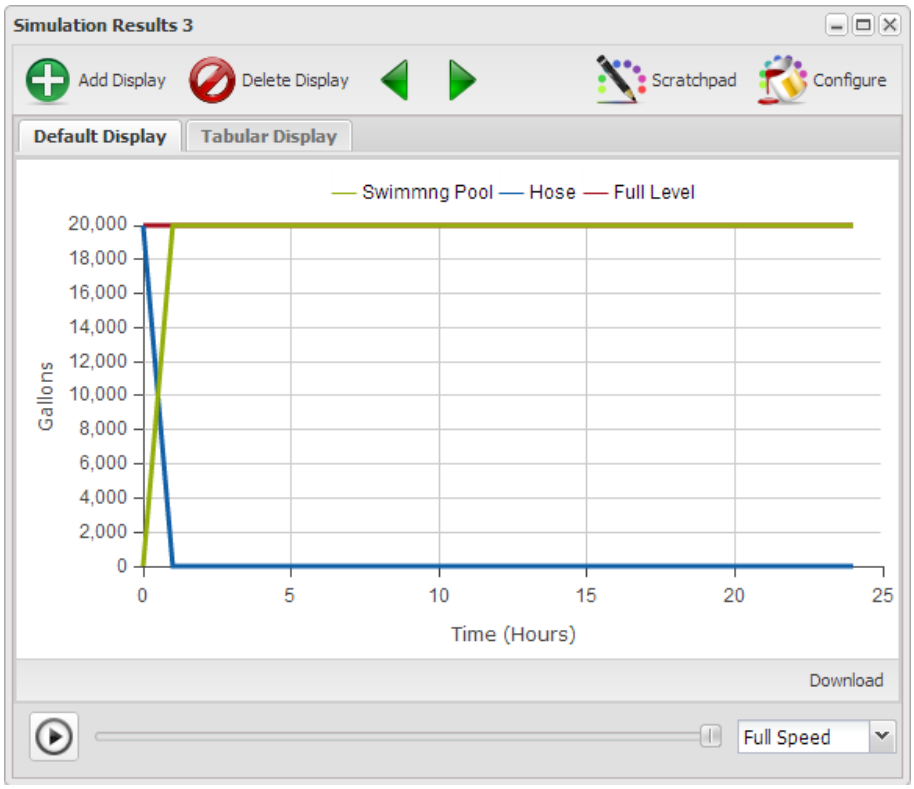


Figure 19. How long to fill the pool

Modeling Note

Hopefully you come to understand that when your models don't do what you expect them to do it's not a problem – it's an opportunity for learning. This is the real reason why we do modeling - to understand and learn. Just think of it as, the more things don't go the way you expect them too, the more opportunities you have to learn.

If you look back at the formula for the Hose, notice it didn't take into account the initial statement that the Hose could only deliver 50 gallons per hour. And, might it be useful if we could see what happened with different Hose capacities?

Figure 20 is a revised version of the model with Hose Capacity as a variable so you can set the capacity of the hose before you run the model.

The new formula for Hose takes into account both the current amount of water in the Swimming Pool, Full Level and Hose Capacity

IfThenElse([Swimmng Pool] < [Full Level], min([Full Level]-[Swimmng Pool],[Hose Capacity]), 0)

With Hose Capacity = 50 if you run the model it should produce Figure 21.

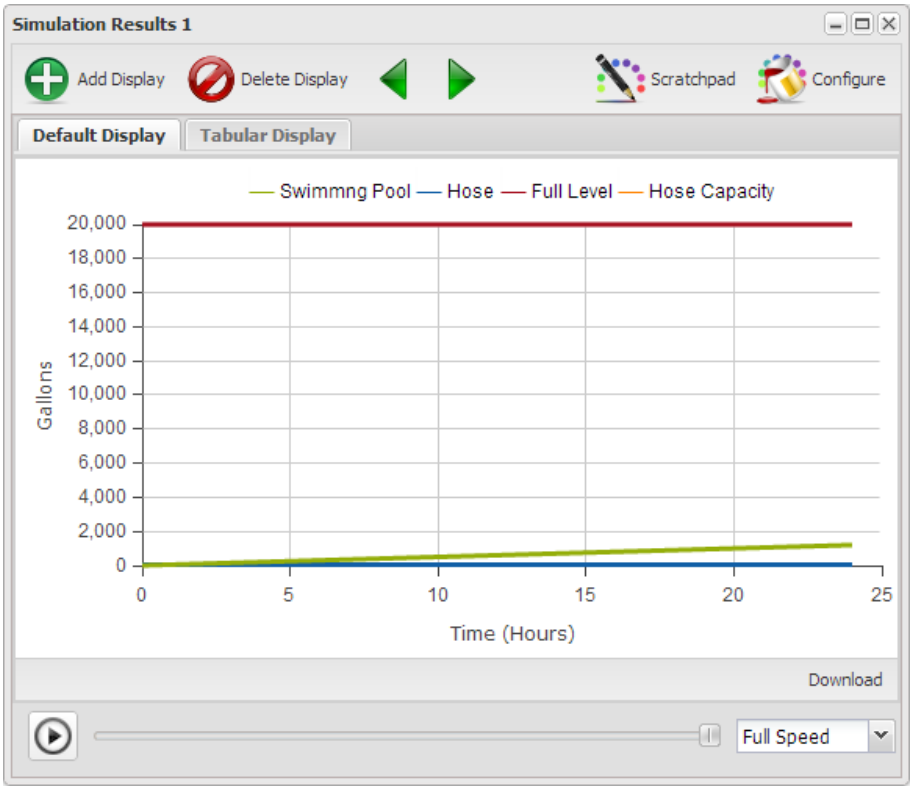


Figure 21. How long to fill the pool at 50 gallons per hour

Was this what you expected? Probably not. Over a period of 24 hours we've not even come close to filling the Swimming Pool.

Open the Time Settings and set the Simulation Length to 600 hours and Run the model again. Your run should produce the an equivalent of Figure 22.

Figure 22 indicates we need to wait 400 hours to fill the pool. That's a little over 16.5 days. I think we need a bigger hose.

While there are a number of things we could do to improve the model at this point I think we've gone far enough with this one.

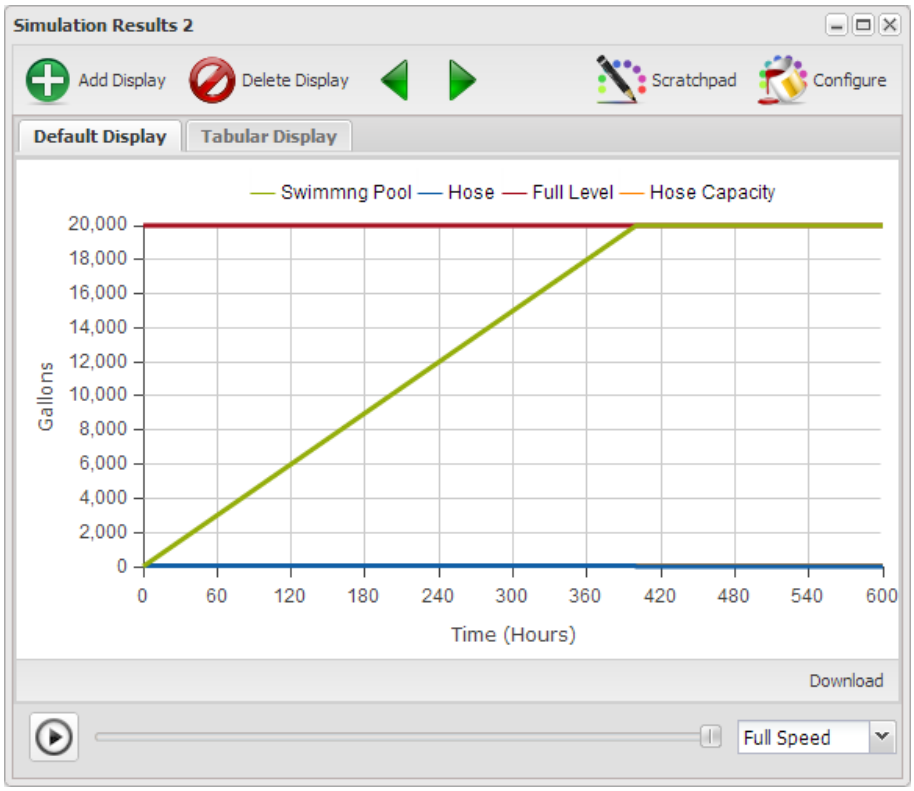


Figure 22. Filling the pool takes how long?

Exercise

Do a number of runs for the model in Figure 20 with different values for Full Level and Hose Capacity. Do you get a sense of how your choice of values impacts the results that appear over time?

Similar Structures / Different Behavior

If you compare Figures 7, 14 and 20 you should find them to be quite similar. And yet the behavior of the models are distinctly different.

Figure 23 presents the previous three models in a general form. This is so you can compare the different behavior of structures that are very similar. Flow Rate, Seeking Factor and Growth Factor are each factors which govern he rate of flow. Goal is a target value which the Growth model doesn't have. The

difference that makes a difference is what happens in the connection between the accumulation, or stock, and the flow.

The link between the stock and the flow provides information from one point to the other and is generally referred to as feedback, mostly likely because the information travels in the opposite direction as the flow.

Linear

In the Linear model the Flow simply depends on the Flow Rate variable, which is expected to be some constant value. This model is referred to as linear because the Accumulation of Stuff is a straight line as you can see in Figure 24.

Balancing

In the Goal Seeking model the State Change depends on the difference between the Goal and the Current State. This difference influences the State Change to increase the Current State until it reaches the Goal. The structure tries to bring about a balance between the Current State and the Goal so the difference is zero, and then there's no more State Change.

Reinforcing

In the Reinforcing Growth model Added depends on the value of Reinforcing Accumulation. This influences Added to increase the Reinforcing Accumulation which increases Added. One might consider a Reinforcing structure to be a Balancing structure that's out of control.

Would you believe that no matter how complicated a model may look it's really only some number of these structures connected together? In the next chapter you will begin actually building some models and investigating the implications of these structures.

Exercise

The values in the Figure 23 model elements were contrived so when you run the model it will produce the graph in Figure 24.

- Can you figure out why the values assigned are responsible for the curves produced?
- Alter the values for the parameters in the Configuration Panel and run the model to get a sense of the impact initial values have on the behavior of these structures.
- Can you explain to someone else the difference between Linear Growth, Goal Seeking and Reinforcing Growth in terms of why the structures produce the behavior they do?

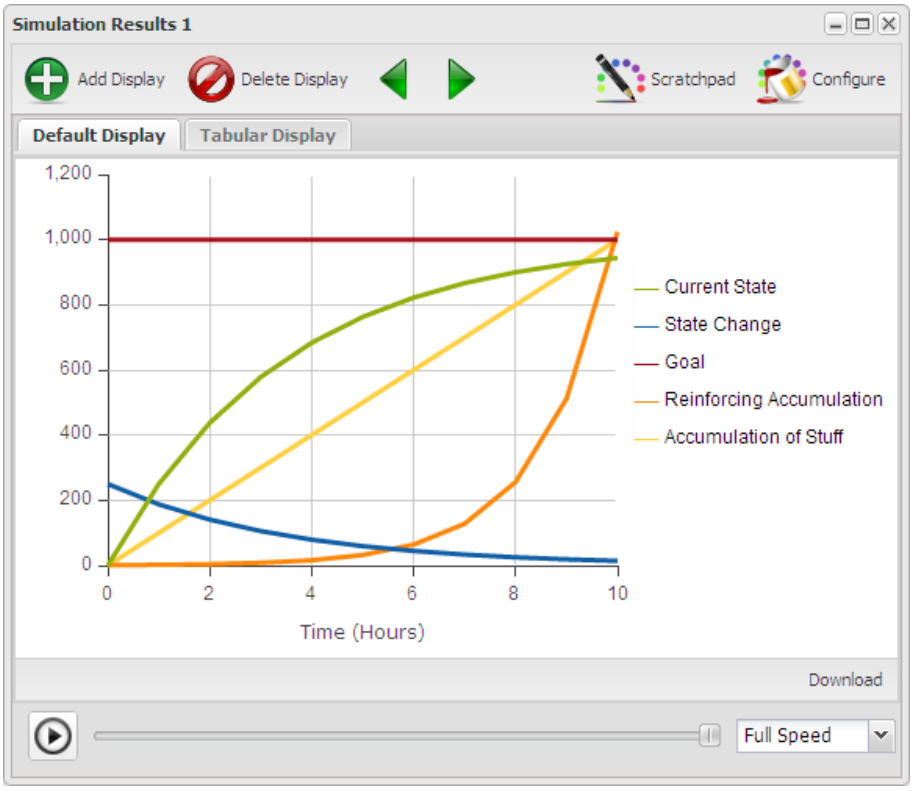


Figure 24. Balancing and Reinforcing Results

Summary

- Models are simplified versions of the world around us.
- We build models to help us understand and learn.
- We build simple models and add to them as we learn with them.
- Building models and learning is an iterative process.
- We learn as we go and seldom do we get models right the first time.
- Reinforcing and Balancing structures are the basic building blocks for all models.
- These building blocks can aid in understanding aspects of our interactions with the world around us.

Chapter 3

Chapter 2 - Dynamic Building Blocks v2 13.05.31

When we build things our skill in using the tools we work with has a very definite impact on things we create. Figure 1 is the last model from the previous chapter. In this chapter we hope you will become far more familiar with, and comfortable with, the Insight Maker environment, the creation of these structures, as well as a deeper understanding of why these structures for the building blocks of everything else you will create in the Insight Maker environment to further your understanding and learning.

The Blank Canvas

When you create a New Insight you don't actually have to start with a blank canvas. Insight Maker presents you with a very simple working Rabbits Population model. This is just so there's something there to work with when you first start. If you click the **Clear Sample Model** button you will then have a blank canvas on which to create. In the next few segments you will learn how to create the three basic structures from which all models are constructed.

Notice in Figure 2 that similar tools are grouped together on the **Toolbar**. Only a portion of the **Toolbar** is displayed though it's enough for what will be covered in this section. If you wanted to see the rest of the **Toolbar** clicking the double caret at then end of the **Toolbar** will display the rest of the items on a drop down.

To use any of the **Primitives** click on the icon on the **Toolbar** to select it, then click on the canvas where you want the item located. For each tool there are a set of allowed uses. Once you place the item on the canvas it is named for what it is, with that name selected so you can type in the name you want. Names can contain any characters except braces "{}", brackets "[]", parentheses (), and quotes '. If the label is not selected you can double-click it to select the label

and then enter a new one, or you can enter the label in the **Configuration Panel** though we'll address that in a bit more detail later.

Exercise 2-1

Practice placing [**Stock**] and [**Variable**] [**Primitives**] on the blank canvas in Figure 2 and naming them. You can remove a [**Primitive**] by clicking on it to select it and then pressing the **Delete** key or clicking the **Delete** button in the **Actions** section of the **Toolbar**. Note that the **Save** option is disabled so you won't be able to save what you create. **Note**: This is only for the review copy. In the final electronic copy you will be able to save what you create.

Stocks, Flows, Variables and Links

[**Stocks**] and [**Variables**] are connected to other [**Stocks**] and [**Variables**] using [**Link**] and [**Flow**] [**Connections**]. The rules for connections are very explicit because Insight Maker has to figure out how to simulate the model. The allowed connections are depicted in Figure 3. The next chapter will present several types of models where the rules for connections aren't nearly as rigid.

If you select **Link** from the **Connections** segment of the **toolbar** then hover over a model \p{Primitives) on the canvas a small arrow pointing to the right shows at the center of the \p{Primitives). If you select a **Flow** the small arrow will only show up over a [**Stock**] as a **Flow** can only connect to a [**Stock**].

Center the **cursor crossing double arrows** over the right arrow, which should then change to a **pointing finger hand**. Drag the mouse over to a second model element and the arrow tags along while the **Connections** is drawn. If neither **Link** or **Flow** is selected then there will be no right pointing arrow when you mouse over the primitive. We'll go into more detail about connections shortly.

Exercise 2-2

Click on the **Unfold Model** button in the lower left corner of Figure 3, and then repeatedly click the **Step Forward** arrow to walk though an unfolding of the Valid Primitives model while you read the comments on the lower bar.

Note the setup takes a few seconds to please be patient. This will only be the case in the review copy, not the final eBook.

Hopefully the rules associated with the connections were easy to understand. Just remember that Flows represent the movement of stuff while Links only communicate the value of something from one location to another.

Valid Primitive Connections

The valid primitive connections of Figure 3 are described as follows.

Flow

A Flow adds stuff to a Stock, subtracts stuff from a Stock, or moves stuff from one Stock to another. The only way to change the quantity of stuff in a Stock is with a Flow.

- A flow out of a stock decreases it. If where the flow goes isn't relevant to the model then it just flows from the stock to the canvas. Select Flow from the toolbar and then click on the arrow that appears on the stock when you mouse over it and drag onto the canvas and release.
- A flow into a stock will increase it. If you don't care where the Flow is coming from then you first have to draw the Flow from the Stock to the canvas and click the Reverse button in the Connections section to get the Flow to come into the Stock from nowhere. It's just a quirk of the web implementation.
- A flow from one stock to another decreases the source and increases the destination. The get a flow between two Stocks draw the Stocks first and then draw the Flow from one Stock to the other.
- Flows can be bidirectional and we'll talk more about that the first time we use one in a model.
- Flows take time! Please remember this.

Link

A Link is used to communicate a value from one element to another. There is no flow of stuff through the link itself. The communication is considered to be instantaneous.

- You can use a Link from a Stock to a Variable to communicate the value of the Stock to be used in an equation. This does not change the Stock.
- You can use a Link to communicate the value of a Stock to a Flow to be used in the equation determining the value of the Flow in the next iteration. The Link does not change the value of the Stock.
- You can use a Link to communicate the value of a Flow to a Variable to be used in an equation. This does not change the value of the flow.

- You can use a Link to communicate the value of a Variable to a Flow to be used in the equation that defines the flow. This does not change the value of the Variable.
- You can use a Link to communicate the value of a Variable to another Variable so that value can be used in an equation in the destination variable. The link does not change the value of the source Variable.
- You can use a Link to communicate the value of a Variable to a Stock to be used as it's Initial Value when the simulation begins. The value of the Variable is computed and assigned to the Stock as the simulation begins and it has no influence on the Stock during the simulation.

When you draw a link from one element to another it is created as a straight line. There are times when you would prefer that the connection be other than a straight line to make the diagram easier to follow. You can turn a straight line into a multiple segment line as follows.

- Click on the link to select it.
- Hold down the shift key and click somewhere in the middle of the link then release. This puts a little node on the line.
- Click on the node and move it as you wish to create a two segment link.
- You can create as many segments as you need, simply repeat the second step above.
- If you wish to remove the segments select the head of the link, move it off the element it's connected to and then reconnect it. It will now be a straight link.

Exercise 2-3

Go back to Figure 2 and recreate Figure 3 for yourself and as you create each element think about what that particular element is for. Actually making the connections helps develop a level of skill and comfort which will serve you well in the future.

Configuration Panel

Each of the four elements used to build a model has some of the same configuration options though because each has a different function there are some unique configuration options for each item. Some of the most frequently used options will be described in the following sections. The ones not described here will be described the first time they are used.

The Configuration Panels for Stock 1, Flow 1 and Variable 1 form Figure 3 are displayed in Figure 3a and are described below.

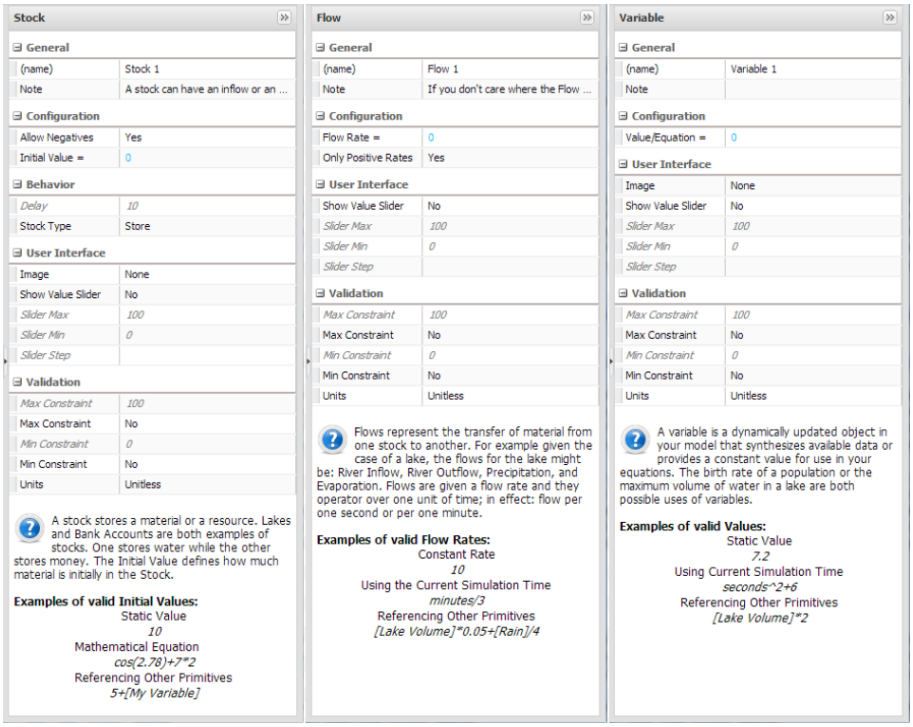


Figure 3a. Stock, Flow & Variable Configuration Panels

General

This section is where you can assign the (name) and Note for an item.

- **(name)**. This is the label that you see on the item. You can double-click the item on the canvas and edit the label on the item itself or change it here in the configuration panel.
- **Note**. Here you can enter a description of the item. You can enter short descriptions directly into the field. If you click the down arrow in right of the field it will open the **Note Editor** dialogue window which allows some formatting options. The note that you enter here will pop up when you mouse over an item and click on the little **i** that appears on the item. If the element of the model is selected you can also open the Note Editor window by CTRL+‘(Control+Backquote). Adding comments to a model helps others to understand what you were thinking and when you go back to the model in the future the comments will help you understand what you were thinking when you put the model together. Yes, you completely

understand now, though will you remember next week, or a year from now?

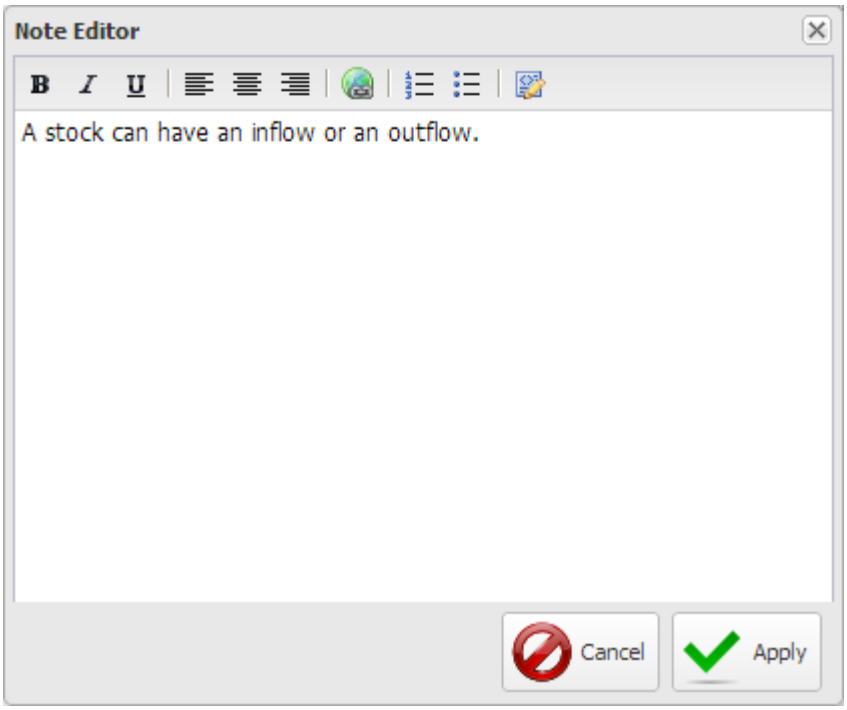


Figure 3a. Note Editor

Configuration

This section is used to define how the element behaves during the simulation and is a little different for Stock, Flow and Variables, though quite similar. The behavior is essentially controlled by an equation which is defined in terms of the variables connected to it. This is an initial value for a Stock. You may enter a short value into the field though if you click the down arrow in the right of the field the **Equation Editor** window will open. In this window you can define the formula that defines the behavior of the element. You can also open the **Equation Editor** for an element by mousing over the element and clicking on the **equals (=)** sign that appears. All the built in functions on the tabs at the bottom of the window have descriptions associated descriptions and examples.

Additionally in this section you define whether stocks can have negative values and whether flows can flow in both directions. We'll talk more about these options the first time we use them.

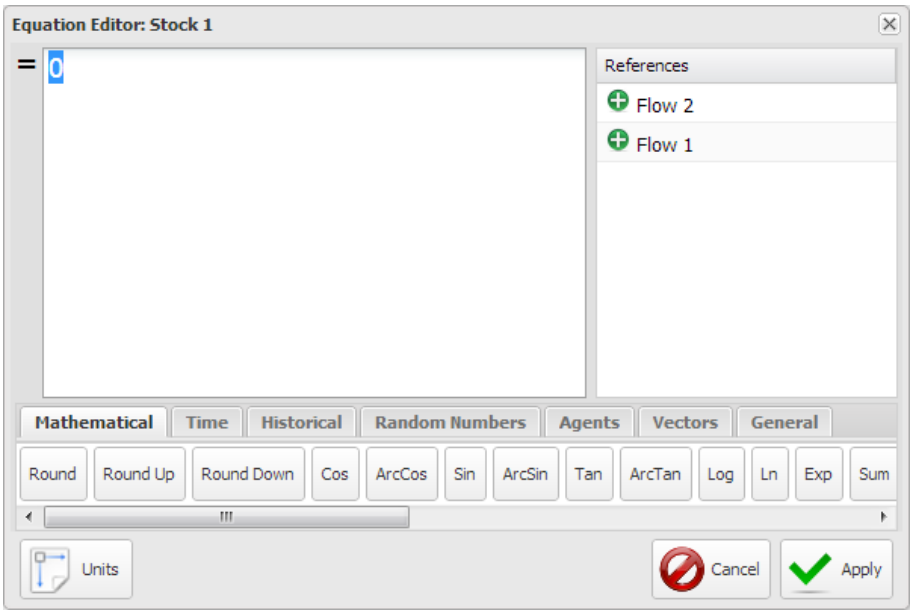


Figure 3b. Equation Editor

User Interface

It is in this segment of the configuration panel that you define a slider for an element, if there is to be one. You can define a sliders for Stocks, to define it's initial value and for Flows and for Variables. Once you indicate there is to be a slider you then define the maximum and minimum values it may have, as well as the step size, how small are the variations allowed. If you leave the step size field blank then the slider can vary continuously.

An element may have a slider or a formula though not both. Sliders override equations. If you enter an equation and it disappears check to see if there was a slider defined and it hasn't been turned off.

Validation

This section allows you to indicate if there are Maximum and/or Minimum constraints on a Stock, Flow or Variable.

Additionally it is in this section that you assign the Units for an Stock, Flow, and Variable. Units are very useful in helping to ensure the soundness of a model. Units will be covered extensively in Chapter 4.

Common Property # 1

To this point you've learned how to develop a static picture of a model. It is actually a model and provides a sense of the relationships between the various elements. What it doesn't give you a sense of is the dynamic nature of these interactions over time. What are the implications of the relationships? In the next few sections you'll learn how to bring your model to life.

Look at the pictures in Figure 4 and ask yourself what it is that these images have in common. The images all represent very different kinds of things, some living, some not, though there is a characteristics they all have in common. Have you figured it out?

Figure 4. Common Property # 1

Maybe you notice the rabbits from the previous chapter? The things depicted in the various images all grow in one way or another, and some faster than others.

Constructing a Growth Structure

Lets use Figure 5 to construct a basic growth structure and in the process you'll learn about several of the parameters associated with the different elements of a model.

- Click **Make New Insight** on the **Toolbar** and then click the **Clear Sample Model** button.

- Place a Stock on the canvas and label it Stuff.
- Now make sure the Stock is selected and take a look at the Configuration Panel on the right.
- Click on Flows to select that element.
- Mouse over the stock and click when the arrow appears at the center and drag onto the canvas somewhere outside the stock. Which direction doesn't make a difference though make sure you're a couple inches outside the stock before you release the mouse button.
- While the flow is still selected click on Reverse so you have a flow into the stock.
- Notice that the parameters in the Configuration Panel are different from those for the Stock.
- Click in the field to the right of Flow Rate = and change the zero to a 1.
- Click the Run Simulation button, which is on the ">>" drop down on the right end of the toolbar. You've now successfully created, and run, your first model. Admittedly it may not be very exciting though it is the first one of many to follow and it runs.
- Save this as you will add to this model shortly.

Notice that the model ran for 20 years. That's because we used the default Time Settings.

Exercise 2-4

Open the Time Settings dialogue associated with Figure 5 and setup and run the model for different values of Simulation Length and Time Units. In what way do your changes alter the output?

Try creating a slider for your stock and flow. Set them for different values and run the model. The idea is just to develop your skill as well as a level of comfort in working with the tools.

Notice that in each case what you get is a rising slope for different time periods and at different angles. What you perceive in the graph is referred to a linear growth though this growth doesn't actually represent the growth associated with the images depicted in Figure 3. In those situations growth at each time is actually dependent on the size of the accumulation or stock at that time.

If we evolve the model of Figure 5 into Figure 8 so the flow is dependent on the amount of stuff we find the resultant growth to be very different.

Figure 8 represents only a couple of changes from Figure 5 as follows.

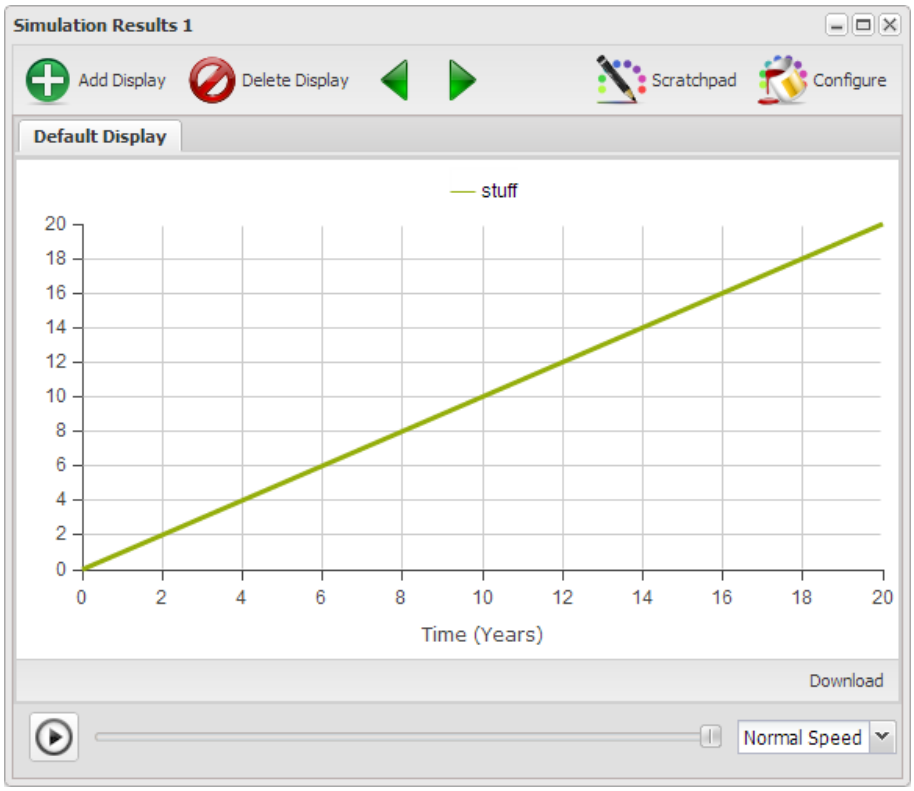


Figure 6. Your First Model Output

- Extend the model you created based on Figure 5 as follows.
- Connect a link from stuff to Flow and reshape it with a couple of handles to improve visibility.
- Mouse over the Flow and click on the = sign to open the **Equations Editor** and set the Flow to

  $$stuff$$

  as in Figure 9. You can do this just by clicking on **stuff** in the **References** list.
- Open **Time Settings** and set the **Simulation Length** to 10.
- Now Run the model. Note that because of the width of the embedded model you can't see the whole **Toolbar**. Clicking the » just to the right of the **Tools** section of the **Toolbar** the rest of the options will drop down and you can select the **Run** option. You should now see the diagram in Figure 10.

The result of the run from the model in Figure 8 is depicted in Figure 9. The value after 10 Years is 1,024 which you should realize is just 2^10 as expected

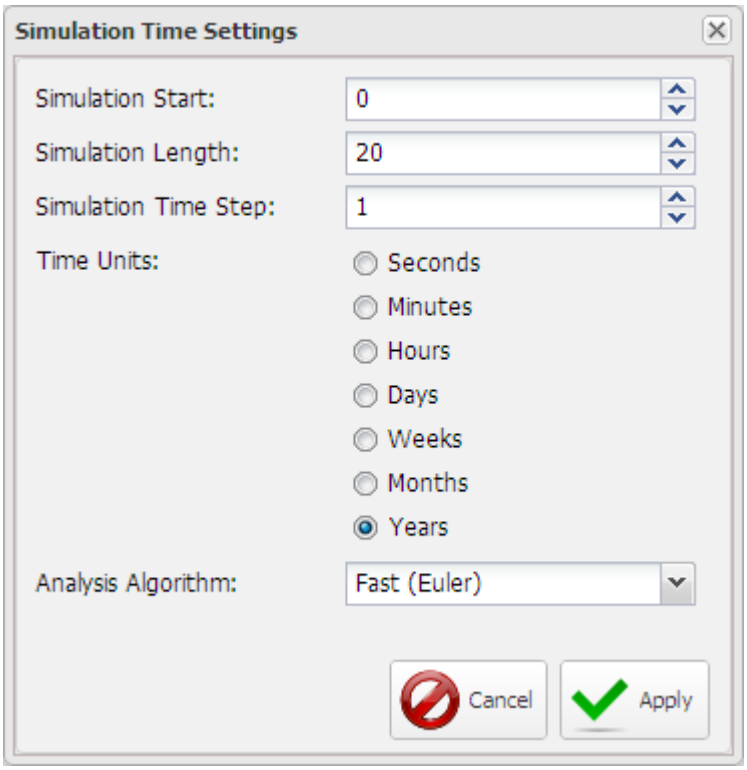


Figure 7. Simulation Time Settings

because we started with a value of 1 and doubled it every year. This curve is referred to as an exponential growth curve.

Exercise 2-5

Notice that the curve in Figure 10 is a bit choppy where it turns up. Run the model in Figure 8 with a Time Step of .5, .25, .125, .0625 and compare the results. What questions are raised by the the results?

Time Units and Time Step Selection

The **Time Units** and **Time Step** selected for a model should be consistent with the time frame and level of detail of the model. You probably wouldn't develop a model about filling a bathtub with water and use **Time Units** of

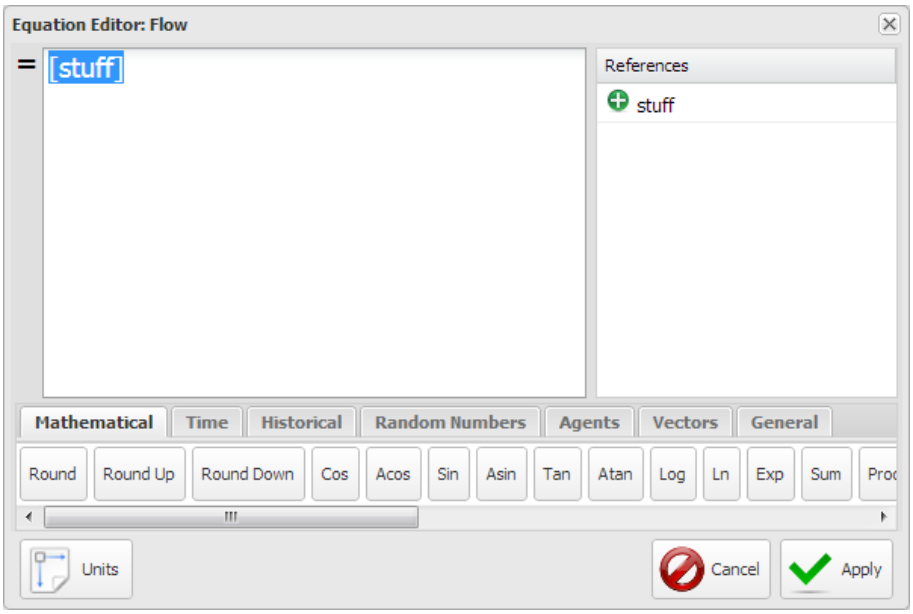


Figure 9. Equations Editor:Flow

months. Minutes are probably more appropriate for this model. The **Time Step** is then selected to ensure none of the relevant transitions associated with the dynamic nature of the model are missed. A **Time Step** of .25, meaning 15 seconds, is probably sufficiently small to ensure there are no transitions missed.

Trial is actually the most appropriate approach to determine if you have an appropriate value for **Time Step**. If you think .5 is appropriate then run the model with 1, .5, and .25 and if the results for 1 and .25 don't differ from .5 then you're probably OK. If .25 produced a different result then compare the .25 result with the .125 result. Once you get two runs where the values don't change then use the larger one.

Given this guidance how would you interpret the results you experienced in Exercise 2-5?

References

- How does DT work? from isee Systems
- DT Situations Requiring Special Care from isee Systems

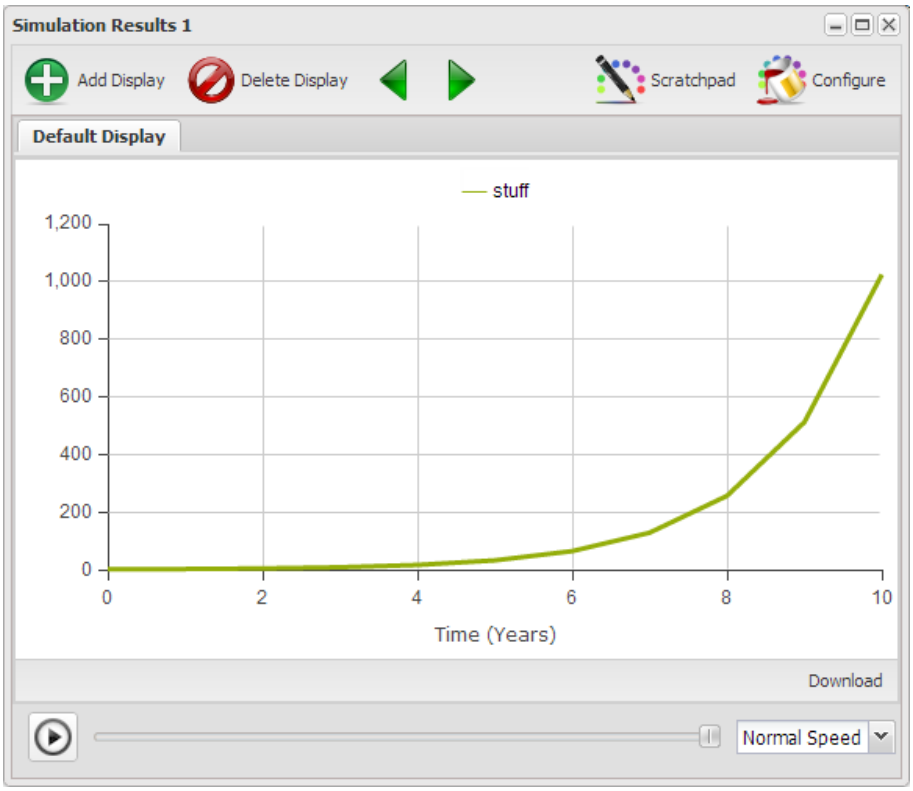


Figure 10. Growth of Stuff

Exercise 2-6

Consider the images in Figure 4 and think about what **Time Units** and **Time Step** you would use in a model representing the growth in each of these areas.

One aspect of trying to model the contexts of Figure 4 that should have become apparent is that there is a piece of the model that's missing.

The model in Figure 11 adds a factor, which is allowed to vary between 0 and 1, which is simply used to govern the flow. Mouse over the Flow and click the equal (=) sign to view the formula governing the flow.

Flow = [stuff] * [factor]

Exercise 2-7

Use the model in Figure 11 to implement the models in Exercise 2-7. Does this structure allow you to construct more realistic representations of the growth situations presented in Figure 4?

Exercise 2-8

The model in Figure 11 is the model for a Savings Account that is defined as compounding annually, i.e. calculating and adding interest once a year. This means that the most appropriate **Time Units** is years with a **Time Step** of 1. There are no other transitions in this model that need to be accommodated. If you run this model with any **Time Step** other than 1 it will result in a less accurate result. What does this happen?

The model of Figure 11 is the standard reinforcing growth model depicted in Figure 1 at the beginning of this chapter. In the process of arriving this model the linear growth model of Figure 1 was developed first, and then evolved. Hopefully through the exercises to this point you have gained a deeper understanding of how this structure works and the extent to which it may be applied to various situations.

Common Property # 2

Look at the activities depicted by the images in Figure 12 and ask yourself what it is that these activities have in common. The images represent very different kinds of activities though there is a characteristics they all have in common. Have you figured it out?

Each activity depicted in Figure 12 represents the pursuit of some goal or objective. Admittedly the goals are very different and each is pursued in a very different manner.

Constructing a Balancing/Goal Seeking Structure

As we have done repeatedly to this point we begin with a linear model consisting of a flow and a stock, along with a flow rate variable. To this we simply have to add a goal and the appropriate feedback and we end up with the model in Figure 13.

When you look at Figure 13 admittedly we added Gap which we haven't addressed before. This was done so we could explicitly plot the difference

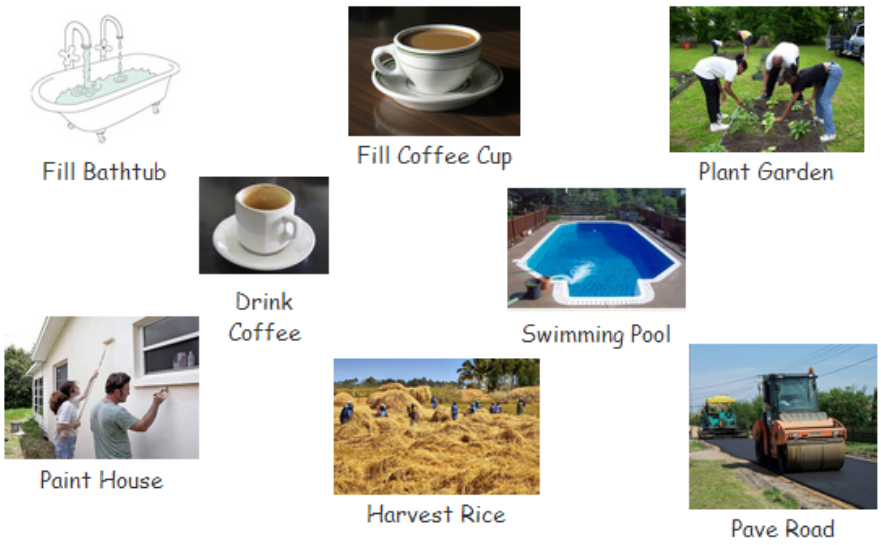


Figure 12. Common Property # 2

between the Current value and the Goal. The factor is simply a multiplier between 0 and 1 to govern the extent to which the Gap governs the change.

Gap = [Goal] - [Current]

change = [Gap] * [factor]

Take a look at the Time Settings for Figure 13 and you'll see that the model was set up to run from 0 to 10 with a time step of 1 and a units of hours. These were just selected to create a generic model where you could consider the Goal to be 100% and the other values as having values between 0 and 100%. This way we can consider the implications of the interactions without getting hung up on the actual values.

If you run this model you should get the result depicted in Figure 14. This shows how the value of Current approaches the Goal as the value of Gap declines with a factor = 0.5.

Figure 14 shows that as Current moves toward the Goal the Gap decreases as does the change which is moving Current in the direction of Goal. Once Current reaches Goal the Gap is zero is change. This structure endeavors to remove the tension between Current and Goal, the Gap, to bring a balance to the situation. Figure 14 presents the characteristic behavior of a Balancing/Goal Seeking structure.

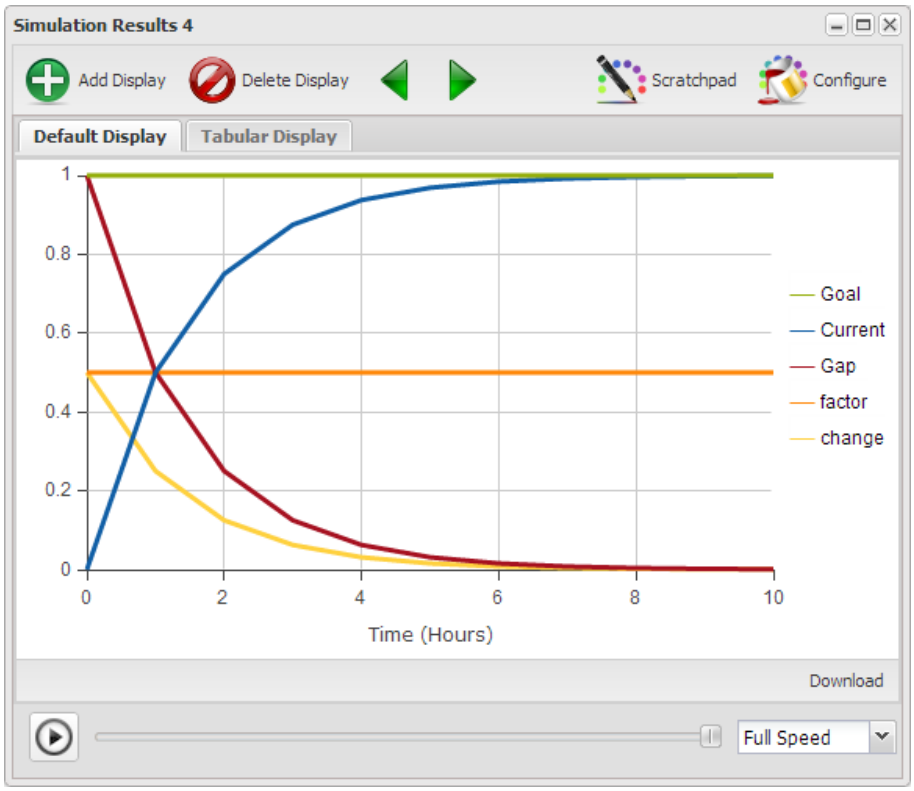


Figure 14. Goal Seeking Result

Exercise 2-9

Run the model in Figure 13 with various values for factor. What do you notice about the relation between Current and Gap? And what do you notice about the curves as the factor gets larger and larger?

Under Time Units and Step Selection we talked about it being essential that the the Time Units were selected appropriate to what was being modeled. In this case since it's a generic model one Time Unit is pretty much as appropriate as any other. The Time Step is another matter though, or is it? We said one chooses a Time Step such that none of the relevant interactions are missed and the change from one Time Step to another doesn't change the result.

Exercise 2-10

Set up the model in Figure 13 to run with Current = 0, Goal = 1, and factor = .75. Now run the model with a Time Step of 1, .5. 25. .125. Does the result actually change? Look at the Tabular Display associated with the Simulation Result. As you make the Time Step smaller and smaller are the results more correct?

Considering that we don't know anything about a real environment being modeled in Figure 13 it's a bit difficult to determine if the result is actually more correct as the Time Step used is smaller and smaller.

You might have also realized by this point that it would be quite difficult if we attempted to use this model to model any of the situations depicted in Figure 12. While progress toward the goal in the situations depicted is promoted by the Gap between the Goal and Current the change in those situations isn't likely to be proportional to that Gap.

Figure 15 presents a modification to the model of Figure 13 where the factor has been replaced by a constraint. It looks like there have been lots of changes though they all cosmetic except the way Workers influence work on a daily basis.

If you run the model with Project Days Work = 60, Workers = 8 and Days Work Completed set at the default of zero and Time Step = 1 you should see the graph in Figure 16.

The reason the graph looks like this is because of the constraint placed on the the work because of the number of Workers available. This is accomplished by the formula embedded in the flow.

> work = IfThenElse([Work Remaining] > [Workers],[Workers],[Work Remaining])

This says that if there is more Work Remaining than there are Workers available to do the work then the amount of work that day equals the number of Workers. This goes on for the first 7 days then on the 8th day there are only four days work required to finish the project which is represented by the different slope on the line on the 8th day. You can see this in detail if you look at the Tabular Display.

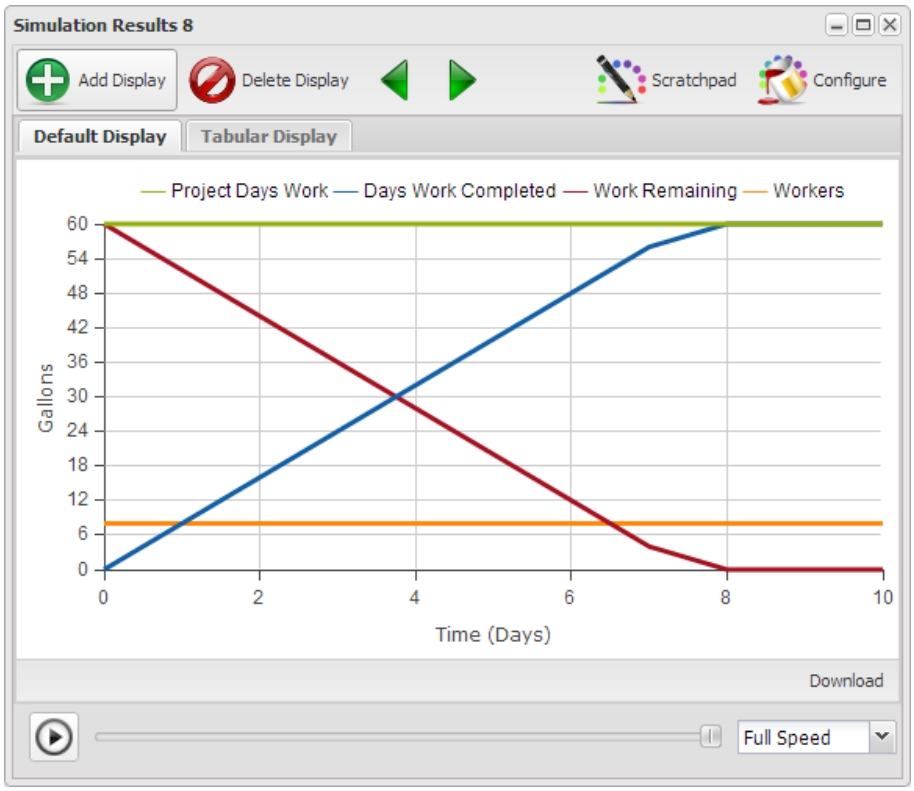


Figure 16. Goal Seeking with Constraint

Exercise 2-11

Set up the model in Figure 16 to run with Time Step of .5. Compare the Tabular Display of this run with the results of the previous run above. By making the time step smaller have we improved the accuracy of result? Why?

Again the appropriate Time Step is one that captures the activity occurring within the model. In this case the Workers are in integers and Project Work days are in integers, and with the Time Units in days the appropriate Time Step is 1. If there were events which happened in the model on the order of hours then you would have to decide whether to alter the model to run in hours or reduce the Time Step to ensure it was small enough so no interactions in the model were missed.

Exercise 2-12

Use the model in Figure 15 and reconfigure it for a couple of the activities depicted in Figure 12. Note that for this exercise you will have to relabel the stock, flow, and variables accordingly. You will also have to decide on the most appropriate Time Units and Time Step to use.

Summary

Hopefully this chapter has helped you become more familiar with the modeling environment and the four model elements you will use most often.

- **Stock**. An accumulation of something that can only be changed by something flowing into or out of it.
- **Flow**. Something moving over time which adds to a stock or subtracts from a stock.
- **Variable**. Constant or equation computed each time the simulation steps.
- **Link**. Used to communicate a value from a Stock, Flow, or Variable, to a Stock, Flow or Variable. The source is not changed and a link to a stock can only be used to set it's initial value.

Because of the nature of the building blocks themselves there are only a small number of valid connections as depicted in Figure 3.

These valid connections are used to create only three different types of structures, linear growth, goal seeking and reinforcing growth. If you are comfortable with these you should be relieved to know that's all there are. Just three simple structures will be used for all the models you will ever build. Of course at times there may be quite a few of these connected together though you should be confident that you know about the pieces.

The models that you have experienced in Chapter 1 and Chapter 2 are referred to as Stock & Flow Simulation Models. These are also referred to as quantitative models because of the values associated with the simulation of these models. In the next chapter we'll investigate a number qualitative models which are also used in developing understanding. These are referred to as qualitative models because there are no numerical values associated with them, though there are times when they can be quite useful.

Chapter 4

Chapter 3 - A Model Is A Model Is A Model v1 13.06.02

In the previous chapters you worked with what is referred to a Stock & Flow Simulation Models. These are referred to as quantitative models because of the numerical values associated with them. While there are other types of quantitative models we won't get into them here. There is also another class of models referred to as qualitative models because they have no numeric values associated with them. There is a level of understanding well served by the use of qualitative models and as such this chapter will serve to acquaint you with several qualitative models that you can create with Insight Maker and find use for in your study of systems.

Models are simplifications of some aspect of the world around us intended to help us understand something. The Stock & Flow Simulation Models are very explicit as they the relations are represented by numerical formulas and the model may be iterated over time. The models presented in the sections of this chapter will present models which may also assist in understanding situations though only from a relationship perspective as there are generally no numerical values associated with the model and they are not iterated over time.

Rich Pictures

A Rich Pictures is a pictorial representation of a set of relationships intended to convey an level of understanding about those relationships and the implication of those relationships. The strength of Rich Pictures is that there are no rules associated with creating a Rich Picture which means you create the picture that helps you understand the relationships.

Figure 1 is a Rich Picture intended to represent the relationships for a Savings Account. It is created using the Picture Primitive connected together with Links.

Picture Primitives are added to the canvas like any of the other Primitives. Just click on the Primitive to select it the click on the canvas where you want the Picture Primitive to be. You don't have to be too specific about the placement as items are very easy to move around on the canvas.

Once the Picture Primitive is placed on the canvas there are several options in the Configuration Panel depicted in Figure 2.

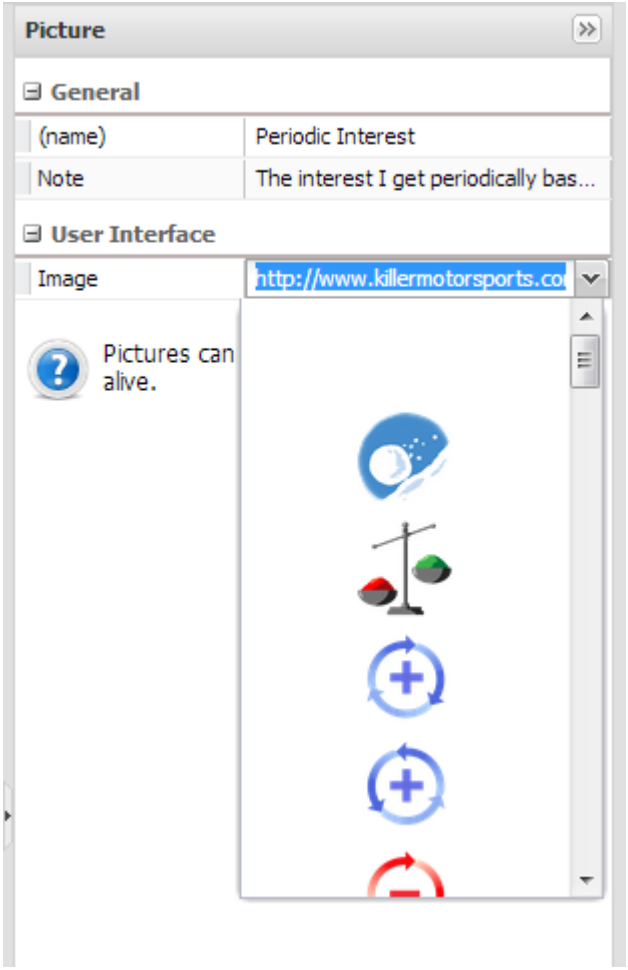


Figure 2. Picture Configuration Panel

The General section of the Picture Primitive is just like the Primitives you've used to this point with a **(name)** and **Note** parameters. The Image option in the User Interface section is new. Here you may select one of the built in pictures from the drop down menu, no picture, the first option on the menu, or paste a URL into the field which points to some graphic on the Internet. Note

that the Internet option only works if you're connected to the Internet.

Once you've created the Pictures simply use the Link element to connect them together as you did in the Stock & Flow models. After you select the Link tool when you mouse over a Picture Primitive the small right arrow will appear in the center as the connection point. Simply click on the selection point and drag to the element you want to connect to. You can't use Flows to connect Picture Primitives.

Note that you can use Stocks, Variables, Text and Buttons in Rich Pictures if you wish and you can add images to them also. Even though this is the case most of the time using Picture Primitives will serve just fine and we'll talk about Buttons somewhere further along the way.

Exercise 3-1

Use the Picture Primitive to add additional elements to the model in Figure 1 and then connect them together using the Link tool. Note that in the Configuration segment of the Configuration Panel for the Link there is an option to indicate whether a Link is bi-directional. Use some of these in your exercise.

Modeling Note

The Text primitive is just similar to any other primitive in that you select it then click on the canvas where you want the Text to appear. Once you create the text item you can edit the text, assign notes to it, connect to or from it with links, and style it with commands form the Style segment of the Toolbar.

In Figure 3 a folder and a Link label have been added to the Rich Picture of Figure 1.

If you select the Link that has the **adds to** label on it you'll notice that the (name) field contains the displayed the **adds to*** label and there's text in the **Note** field. These two attributes are like other Primitives. If the **(name)** field is blank for names **Link** it won't display. Any other text in the field will display on the Link, or near it if the link is segmented and curved. If you segment the Link the program doesn't really know where you want the label to be. As such if you select the link you'll see a small yellow node on the text. You can click on this node and move the label to wherever you would like it to be. Be warned that at times the position is a bit artful though you'll get the knack of it.

Folders are provided so you can enclose a number of items you want to explicitly draw attention to as a concept and have the ability to hide the detail if you choose. To hide the detail click on the little minus (-) sign in the upper left corner of the folder. when you do this the folder closes, displays it's name and the minus (-) sign changes to a plus (+) sign. And you probably also surmised when you closed the folder that you can select an image for a folder or provide an image URL.

Remember the intent of a model is to be some simplification of the world around you to help you understand. There are really no rules for creating a Rich Picture though if you want others to understand it you might want to ensure it is readily understandable rather than confusing. How is the best way to ensure this? Present it to others and have them let you know where they thing it is clear and where they think it is confusing. Then go work on it some more.

In the next section you will be introduced to the Unfolding feature of Insight Maker which you can use to build a script so the model will explain itself to someone else when you're not there.

Figures 4 through 6 provide examples of Rich Pictures for your review. Reviewers please note that these will be replaced with Rich Pictures more appropriate for the target audience.

Causal Loop Diagrams

A Causal Loop Diagram is more structured than a Rich Picture and less structured than a Stock & Flow Diagram. The Causal Loop Diagram was initially invented as a way to express the findings of a Stock & Flow simulation model without having to show the entire Stock & Flow model as it was expected it might overwhelm the audience.

While a Causal Loop Diagram is a qualitative model there is still much one can come to understand from one because of the information presented about the relations as depicted in Figure 7.

The good news is there are no new aspects of Insight Maker you need to learn to create a Causal Loop Diagram. The diagram is created with a Variables, Pictures and Links with Text used to indicate the relationship between the influencing variable and the influenced variable. After you create the variables, which you may represent with the Variable, Picture or Text Primitives you use Links to connect them and identify the relationship represented by the Link.

There are two widely used notations associated with Causal Loop Diagrams, both of which are presented below.

The first notation popularized by Senge in The Fifth Discipline uses an **S** and an **O** notation as depicted in Figure 7.

- **S** implies that the influenced variable changes in the same direction as the influencing variable. If the influencing variable increases the influenced variable increases. If the influencing variable decreases the influenced variable decreases.
- **O** implies that the influenced variable changes in the opposite direction as the influencing variable. If the influencing variable increases then the influenced variable decreases. If the influencing variable decreases then the influenced variable increases.

The influence indicators are created as Text elements and positioned as you deem appropriate to represent the influence representative of the Link.

The + and **-** notation is an older notation and the each has two possible meanings which are deduced from the context of the diagram.

- + implies that the influenced variable changes in the same direction as the influencing variable or the influencing variable adds to the influenced variable.
- **-** implies that the influenced variable changes in the opposite direction as the influencing variable or the influencing variable subtracts from the influenced variable.

The + and **-** notation are considered less likely to generate inconsistencies in a model when there are elements such as production and inventory depicted in Figure 7. In this relationship the + interpreted as **adds to** for production adds to inventory. The situation is such that as production increases inventory increases and as production decreases inventory still increases, just not as rapidly. This should allow you to easily see that the **S** notation might result in a misinterpretation of the diagram as production decreases inventory decreases. To aid in avoiding the confusion some people use Stocks in their Causal Loop Diagrams to indicate those elements which are actually accumulations.

Because creating the Link indicators can get tedious in a hurry after you create the first one it's much easier to hold down the Control Key then click on the indicator and drag to a new location. This quickly creates you a copy of the indicator. You can actually use this sequence to duplicate any element of a model.

As this section is about Causal Loop Diagrams there probably should be some loops defined here somewhere shouldn't there? Figure 8 is a Causal Loop representation of the Balancing and Reinforcing loops presented in Chapter 1 and Chapter 2.

The Balancing and Reinforcing models in Figure 7 vary slightly from typical causal loop diagrams in the following ways.

- **Variables**. Variables in a Causal Loop Diagram are typically just Text though here we've used Stocks for those variable which represent accumulations and Pictures, with no images, to represent the variables. The advantage in using Picture elements for variables is that there are no restrictions on labels because Picture elements are never used in equations.
- **Link Colors**. Because the Link identifiers have to be created separately and are not attached to the Link it's much easier to make + indicators Blue and - indicators Red. This seems to be and evolving convention and then you could just leave the Link indicators out altogether.

While models are a simplification of the world around you intended to promote understanding they also represent an unfolding story. It is appropriate to sequence and label the loops. This gives others an guide to what order to read the story and the intent of the loop itself. The images are created with Picture elements and the images for reinforcing + and balancing - loops, both clockwise and counterclockwise are defined images available from the pull images pull down on the Configuration Panel.

The easiest way to identify whether a loop is a reinforcing or balancing loop is to count the number of minus (-) influences around the loop. If the number is zero or even then the loop is a reinforcing loop. If the number is odd then it's a balancing loop.

Figure 8 presents the two basic structures as Causal Loop/Rich Picture hybrid models.

If the images make it easier for the model to develop and convey understanding then it helps them achieve their purpose. Figure 9 presents a Causal Loop Diagram for The Boy Who Cried Wolf Aesop's Fable.

If you mouse over the various elements of this model you'll note there are notes on each one to help others to understand the model. Yet even with all these aids to understanding it's difficult for others to see it as a story as they're looking at the whole model at once. The Unfolding function in the next section will show you how to overcome this and actually tell a story with the model.

Unfolding a Model

Even though you may conscientiously develop your model and add comments often it turns out that people initially look at the whole model and they're overwhelmed. It might be somewhat like to digest an Elephant in a single bite. Unfolding a model as a story provides a way to overcome this difficulty.

Figure 10 presents a Stock & Flow simulation model intended to be unfolded as a story. Click on the green plus sign in the lower left corner next to Unfold Model, read the text, and then click the Step Forward arrow on the right repeatedly to have the model presented as a story.

This sequencing you just experienced was all done with the **Unfolding** function on the right side of the Tools section of the Toolbar. If you click **Unfolding** in Figure 10 you should see the display in Figure 11.

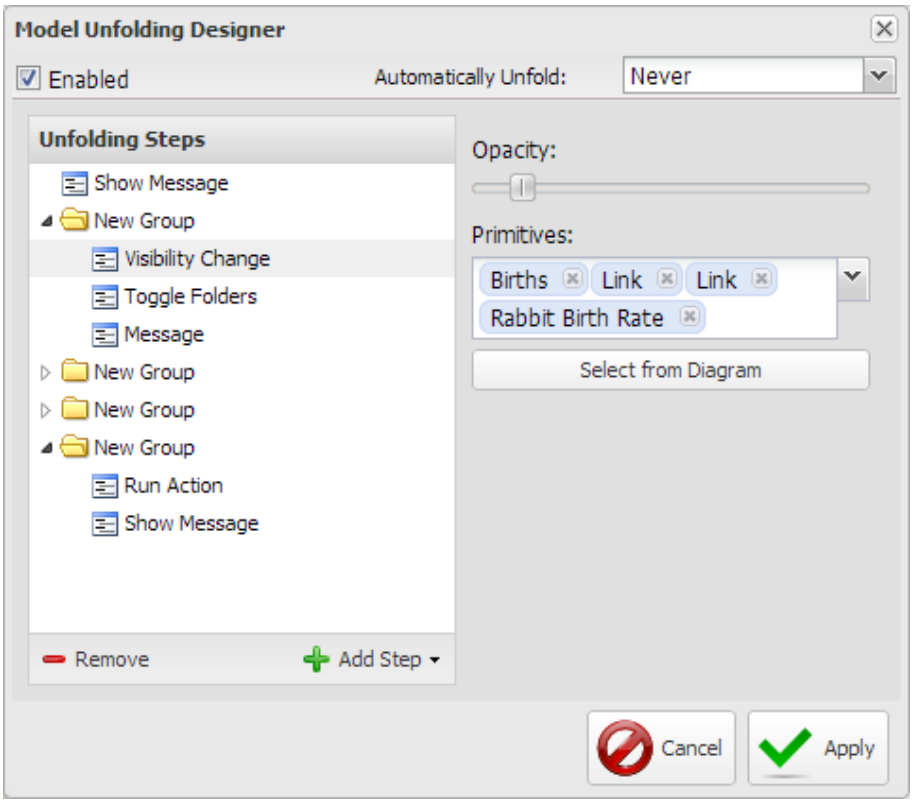


Figure 11. Model Unfolding Designer

The main elements of this window are...

- **Enable Checkbox**. This allows you to actually enable Unfolding. If this box is checked the green plus sign and the Unfold Model will show up in the lower left corner of the window.
- **Automatically Unfold**. There are four options on this drop down allowing you to indicate the conditions under which Unfold Model should execute automatically when the model is opened. The options are Never, For Editors, For Non-Editors, Always.
- **Unfolding Steps**. Lists the steps that you have defined as part of the unfolding. You may reorder steps by clicking and dragging them to the new location. If you click on one of the steps it's definition will be displayed on the right side of the Model Unfolding Designer window. Currently the definition of the first Visibility Change is displayed.

- **- Remove**. To remove an Unfolding Step first click on it to select it then click the **- Remove** button.
- **+ Add Step**. A drop down which allows you to select which type of step you want to add. New steps are added to the bottom of the list and you can then select and drag it to the location in the sequence where you want it. The various steps will be described shortly.
- **Cancel**. Allows you to exit the Model Unfolding Designer and not save any changes.
- **Apply**. Applies all the changes you have made in the Model Unfolding Designer and exits.

The steps you can include in the unfolding are depicted in Figure 12.

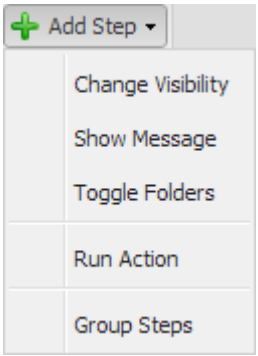


Figure 12. Steps of an Unfolding

As indicated above you can select any one of these steps which will be added to the bottom of the list, then you can move it to the location where you want it. As the following step types are described you might want open the Model Unfolding Designer and click on different Unfolding Steps to visualize how they're defined.

- **Change Visibility**. Allows you to specify the Opacity, with the slider, for one or more elements of the model when that step is executed. The steps may be selected from the drop down or you may select one or more items from the model and then click the **Select from Diagram** button to put those items in the list. Note that clicking the **Select from Diagram** button will replace the already selected ones with whatever you've selected in the model. If you want to add or remove one more more Primitives it's probably better to use the drop down to add or click the existing item to remove it.
- **Show Message**. Allows you to enter a text message you want to be displayed. There are some formatting options available in the Message edit window.

- **Toggle Folders**. Use this option to Expand or Collapse one or more folders. This is really useful if you want to expand a folder, walk through the items in the folder and then close the folder.
- **Run Action**. Provides you with a window in which you can enter Javascript commands to control various aspects of the model. And there are a large number of functions in the Insight Maker API that you can employ in this step.
- **Group Steps**. This step creates a folder in the sequence in which you can place multiple steps. This allows you to indicate there are several steps you want to execute with a single Next Step click. You can see how this was used in the Figure 11 definition of the unfolding for the model in Figure 10. You can open and close a New Group folder by toggling the little triangle to the left of it. Also if you click on a folder you can rename it in the **Name** field on the right.

All new steps are added to the bottom of the list of steps. If you're creating a long series of steps and then you need to add one near the top this may be difficult. In an attempt to provide a work around the Model Unfolding Designer window is resizable.

Figure 13 is the model from Figure 9 above placed here so you can unfold this model and then take a look at the Model Unfolding Designer to see how it was setup. Just click the Unfold

By creating an unfolding story with your model you significantly increase the likelihood that others will understand the insights your model is endeavoring to surface. You might even find that in the process of creating the story you uncover ways to improve or clarify the model for yourself.

Figure 14 is another unfolding that you can walk through and study as to how it was put together. This story attempts to explain why applying pesticides can increase the requirement for spraying more pesticides. It also attempt to distill the concept of Systems Thinking to a single word... And?

Summary

Always remember that a model is a simplification of the world around you intended to help you understand something. You should use a modeling form that adequately sever your quest for understanding. Stock & Flow, Rich Picture, and Causal Loop models are only three of a very large array of possible modeling forms that exist. Use them to the extent that they serve you and when they don't find a form that does. the Modeling and Simulation reference below will provide access to several additional types of diagrams that have been created with Insight Maker, though there are probably more that haven't yet been identified. The Model Thinking Course by Scott E. Page presents a very extensive exposure to many types of models useful for understanding different aspects of the world around us.

Rich Pictures

- Often easier for others to relate to and understand because of the images in the model.
- No rules for creating them though remember that you need to tell a story.
- Because you need to tell a story make the model as understandable as possible.

Causal Loop Diagrams

- Specific Guidelines for how to depict the relationships between the elements.
- Two conventions for expressing relations.
- S and O notation can produce misinterpretations when it comes to stocks.
- The older + and - notation is considered more appropriate.
- Color coding relations is an alternative to both notations, though be consistent.
- Employing explicit stock representations often reduces misinterpretations.
- Hybrid Rich Pictures/Causal Loop Diagrams are often the very meaningful.

Unfolding/Storytelling

- Makes it far easier for others to understand your insights your model is trying to surface.

References

- Senge, P. 1990. The Fifth Discipline: The Art & Practice of the Learning Organization
- Modeling & Simulation with Insight Maker
- Model Thinking Course by Scott E. Page, University of Michigan
- Rich Pictures from Open University Course T552

Chapter 5

Chapter 4 Building a Model v2 13.06.02

```
"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to," said the Cat.
"I don't much care where-" said Alice.
"Then it doesn't matter which way you go," said the Cat.
"-so long as I get SOMEWHERE," Alice added as an explanation.
"Oh, you're sure to do that," said the Cat, "if you only walk long enough."

Lewis Carroll - Alice in Wonderland
```

Now that the most relevant aspects of Insight Maker have been introduced it is appropriate to provide you with a meaningful process and guidelines to use when they set out to build a model to promote an understanding of an area of interest. An aspect of this essential for the development of sound models is the topic of units. While units don't ensure a model is sound, if the units don't match up one can be certain the model is not sound.

Model Construction Process

We develop models to help us understand the implications of interactions, and sometimes guidance. As such, as with Alice above, it is essential that before you begin to build a model you know what it is that you want to understand otherwise how will you know if the model does what you needs to do.

There are a number of guidelines or rules of thumb that you will find helpful when developing a model. These will be presented as Modeling Tips throughout this chapter. The idea is to ensure that the model serves the purpose you started building it for.

Figure 0 presents a generalize concept for of model development.

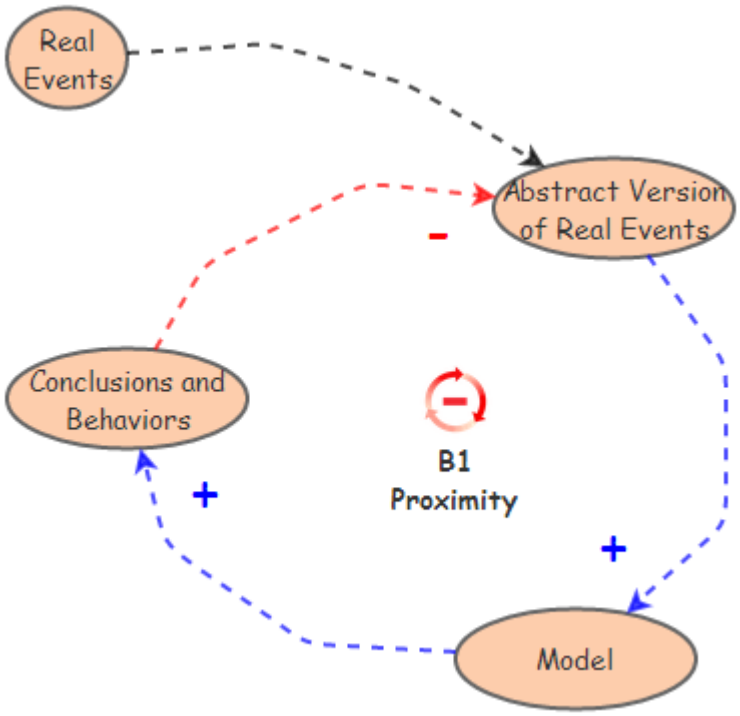


Figure 0. Model Construction Process

The difference between Real Events and Conclusions and Behaviors result in the creation of an Abstract Version of Real Events. The abstraction is then used to develop a Model which promotes a revision to Conclusions and Behaviors. This cycle continues until the Model produces a set of Conclusions and Behaviors which are congruent with Real Events. At this point there's no longer a need to create an Abstract Version of Real Events, meaning you have achieved the understanding you were seeking.

The construction process of Figure 0 is very conceptual and as you continue to develop models you will arrive at a sequence that you are comfortable with an enables you to achieve the understanding your seeking. Figure 0a and 0b present the two model formulation processes presented by Andrew Ford in Modeling the Environment.

In this approach one focuses on the understanding the qualitative dynamics, i.e., problem familiarization, problem definition and model formulation. Not until such time as there is a level of comfort in the understanding of these dimensions, which may employ Rich Pictures or Causal Loop Diagrams, does one progress to the quantitative aspect of model building, i.e., estimating parameters, simulating to explain the problem and sensitivity and policy analysis, which is where the

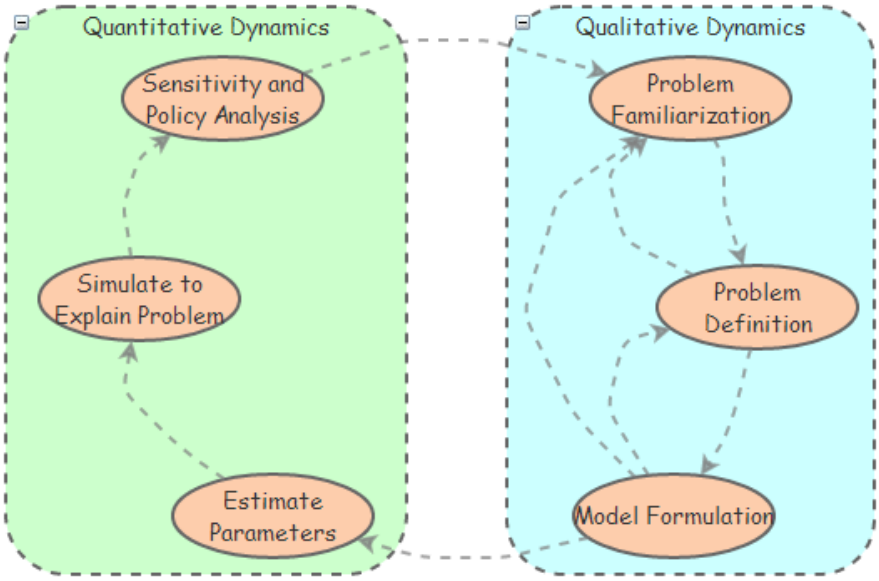


Figure 0a. Emphasis on Model Formulation

Stock & Flow simulation model is employed. The quantitative dynamics may produce sufficient understanding or the process may continue back into the qualitative dynamics area. Model development is an iterative process.

Figure 0b, which may look like complete chaos, emphasizes simulation to provide feedback to provide a better understanding of all other aspects of the modeling process.

Here the believe is that actually simulating all stages of the model are the best way to ground one's understanding of all other aspects of the model development process.

As you develop models you will develop an approach which is probably somewhere between Figure 0a and 0b that you are comfortable with. That is probably the most critical aspect, i.e., that you be comfortable with your process and it make sense to you.

Modeling Guidelines

Figure 0c presents a number of guidelines or rules of thumb it would be good for you to keep in mind as you develop your models. Some of the following are only relevant to Stock & Flow simulations, and which they are should be quite obvious.

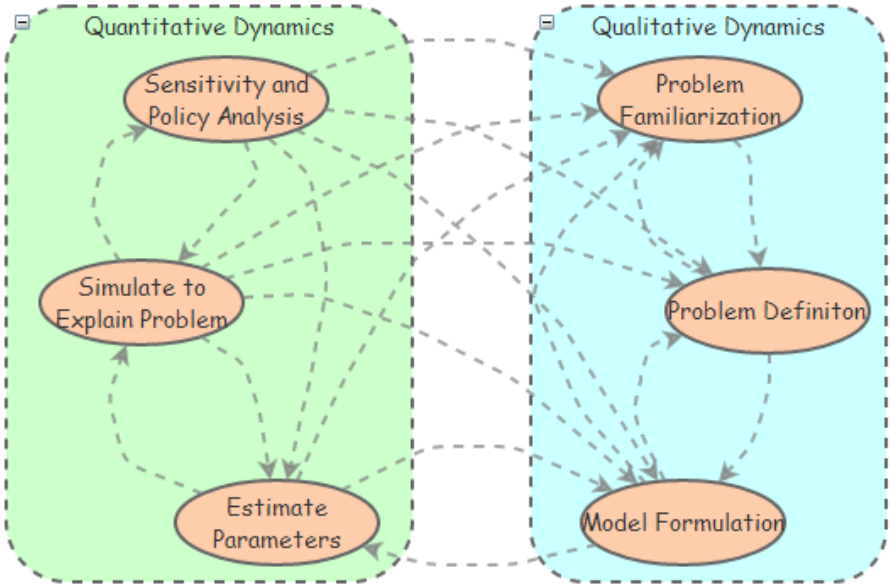


Figure 0b. Emphasis on Simulate Early & Simulate Often

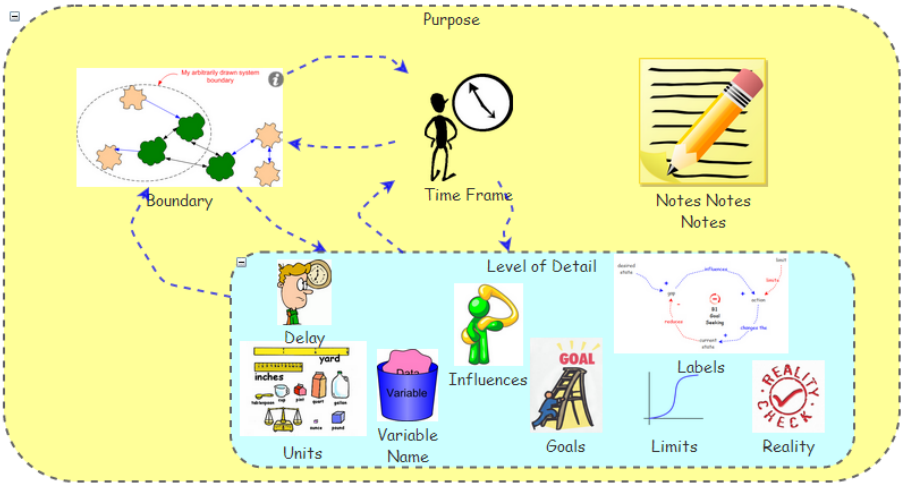


Figure 0c. Modeling Guidelines

Remember that model development and the understanding that comes with it is an iterative process. It's almost impossible to create all the pieces as they should be the first time around. Look at it as do a little, learning a little and repeat.

- **Purpose**. Have a sense of what you want the model to accomplish and expect the thought to evolve as you devleop the model.
- **Boundary**. A boundary allows you to explicitly define what's part of the model and what's not part of the model. If you're unclear on the purpose of the model and unable to establish a boundary how will you ever know when to stop adding things to the model?
- **Time Units**. Will the interactions in this model be depicted over Years, Months, Days, etc. In this case the initial thought is that it will be hours. And you should realize that you initial thought may have to be revised once you begin developing the model.
- **Simulation Length**. How long might the interactions have to be modeled for. Here again the answer may be obvious, or you may have to start with an estimate and revise it after working with the model. Here we'll estimate 4 hrs.
- **Time Step**. Here again you have to estimate a value based on the smallest time of transition you expect in the interactions and then test it to see if you you're close enough. In this case we'll start with 0.5
- **Notes Notes Notes**. As you build your model add notes to the elements so you can refer back to them later to get a sense of what you were thinking when you created them. Yes, you tell yourself you know what you're doing at the moment, though you'll be surprised at what you won't remember a week, a month or even a year from now. Notes also make it much easier for others to understand what you intended when you created elements.
- **Variable Names**. A stock represents a quantity and should be labeled with a directionless noun our noun phrase, you know, a person, place or thing. Avoid directional modifiers such as increasing, decreasing, growing, slowing, etc. as they tend to make a model very difficult to understand. A flow represents something moving over time so it's label should be something one would easily think of as moving over time as walk, speed, flow, etc.
- **Loop Labels**. If you're developing a Causal Loop Diagram or Stock & Flow Diagram be sure to label and sequence your loops so others have a sense in what order to read your story.
- **Goals**. Balancing loops always have goals. Make sure they're explicitly identified.
- **Influences**. Make sure you include all relevant influences and only the relevant influences. Sometimes you include items because you can't figure out if they're relevant or not. That's OK as long as you remember to

later take out the ones that aren't relevant. If you leave influenced which aren't relevant they are likely to result in confusion later.

The following items are most likely relevant only for Stock & Flow simulations.

- **Stocks**. Identify which items are the the stocks, or accumulations, in the model that will change over time. Stocks are often easy to identify if you think about stopping time. When time stops a stock still has a quantity. In this case it's the distance from Grandma's house as Red walks toward it.
- **Flows**. Identify the flows which are responsible for changing the stocks over time. If time stops a flow has no value. In this case it's walking.
- **Delay**. Delays can have very unexpected impacts on the behavior of a model. Where there are delays make sure they're explicitly identified.
- **Units**. Units can be very instrumental in assuring model validity. While consistency of units doesn't guarantee model validity if the units are inconsistent you can be sure the model is not valid.
- **Limits**. If there are limits on Stocks, Variables or Flows be sure they are explicitly stated so Insight Maker can inform you if the model generates out of limit value. This will signal you that there is a problem with assumptions or initial values.
- **Reality Check**. Ensure the model is producing results which are consistent with reality. If it is not then it's an opportunity for learning.

The guidelines are far too much to memorize though if you refer to them as a check list over time they will actually become second nature and you'll find yourself checking them as you're adding elements to a model.

Can Red Get to Grandma's House

Here's a simple example of a question that might be answered with a model. And yes, it is quite obvious you could just do the math though would you get any better at building models if you did?

Little Red Riding Hood want's to know how long it will take her to get to Grandma's house if she walks at 2 miles per hour and Grandma's house is 4.5 miles away thought the woods.

In this statement what is to be figured out is very easy to identify. Sometimes it's not so easy and you have to dig a little.

Figure 1 represents a simple model of Little Red Riding Hood walking to Grandma's house.

While this may look like a rather trivial model there are several aspects of this model that warrant a few notes, and some of them we've not considered before.

Modeling Tips

If you click on the stock and look at the configuration panel you'll notice that the last item in the list, Units, is set to miles. Units were not addressed in the first two chapters as they are so important we wanted to ensure we could focus on them in this chapter. You use units to help ensure that your models are sound. Not that units will not guarantee that your model is sound though it the units don't work out right you can be sure it's not, and Insight Maker checks them for you. Figure 2 shows the Configuration Panel for the stock.

If you click on the flow and look at the configuration panel you'll notice that the units for walk is miles/hour as depicted in Figure 3.

The flow has a units of hours as that's what was set up in Time Settings as the Time Units for the model. All the time settings are showing in Figure 4.

You might now be asking, how the Walk in miles/hour gets turned in to Distance to Grandma's in miles? Because we've selected a Time Step of 0.5 each simulation step multiplies 0.5 hours x 2 miles/hour to get 1 mile traveled each time step. And the units are consistent. Later you can try changing the Time Units and running the model to see how that affects the answers. It's not actually this simple though with a constant flow rate this description is close enough.

Modeling Tip

There are a large number of units predefined in Insight Maker. If you click in the Units field and then click on the drop down on the right the Units Selection window will open as depicted in Figure 5. Here you can select from predefined units, though it's usually easier to just enter the appropriate units into the Units field. There is also a way to define Custom Units thought we'll cover this option in a later chapter.

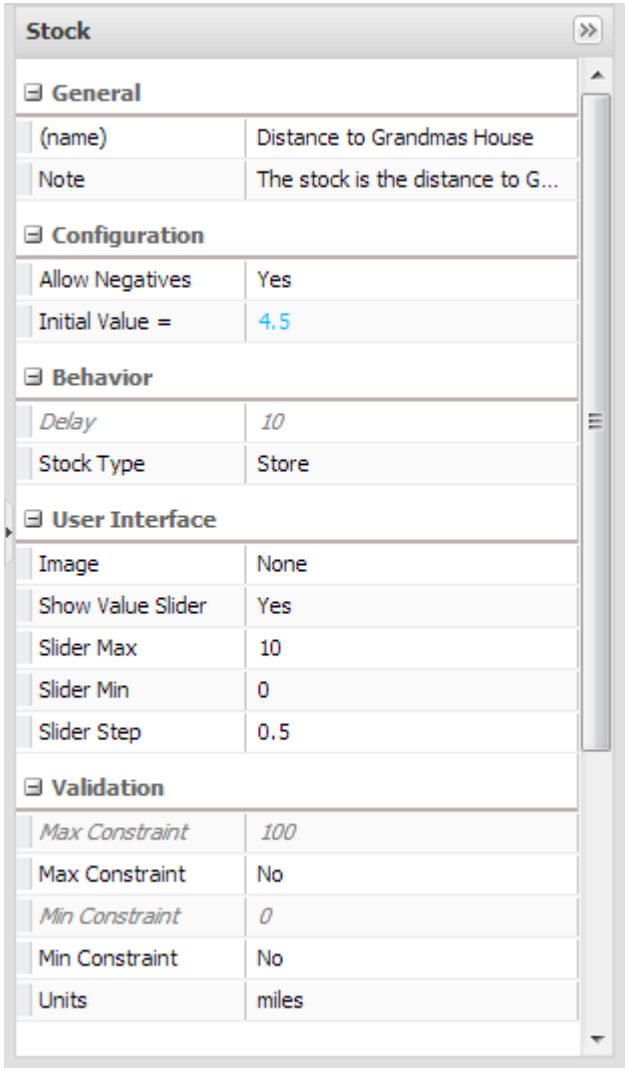


Figure 2. Units for Distance to Grandma's House is in miles

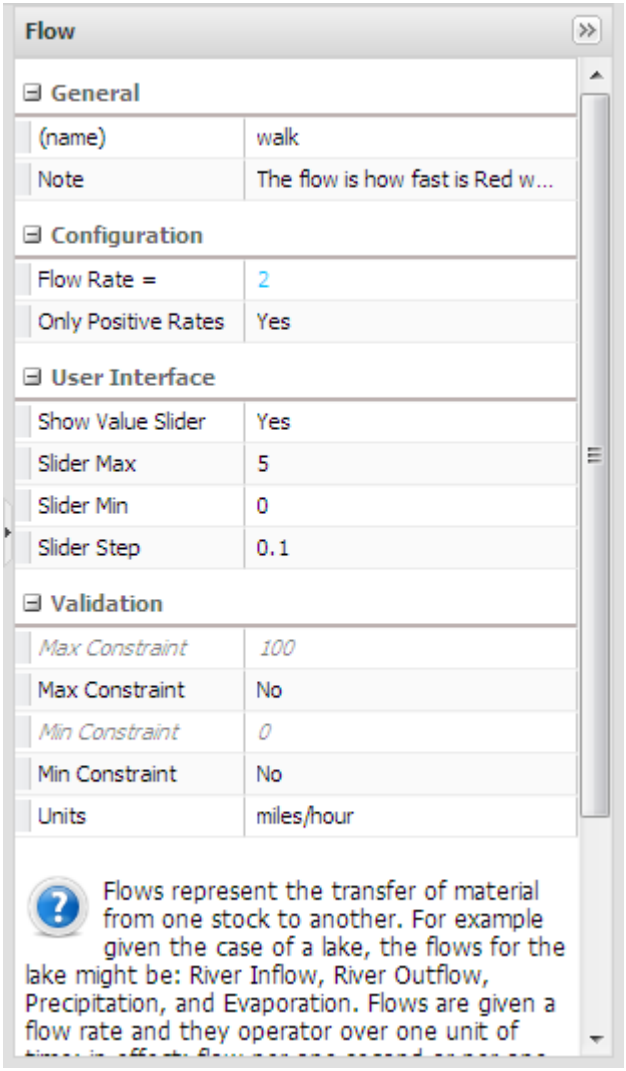


Figure 3. Units for Walk is in miles/hour

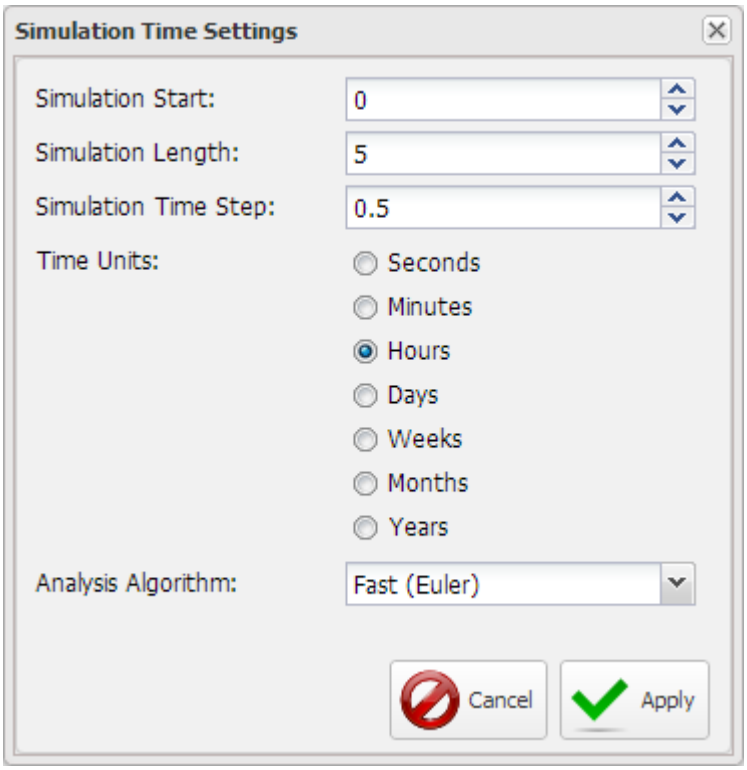


Figure 4. Time Settings for the Walk to Grandma's House

Now click the Run Button to simulate the model. The Run Button was added as a convenience because the Run button on the menu bar is off the right side at the top. The Run Button in the model will probably go away in the final version of this writing.

From the graphic results in Figure 6 and the tabular results in Figure 7 is should be evident that there are some enhancements that need to be made to our Walk to Grandma's House model.

In both the graphic display and tabular display it's evident that Red didn't stop when she got to Grandma's house, and one might wonder where she ended up after 5 hours of walking. And from the tabular display at 2 hours Red was 0.5 miles from Grandma's House and at 2.5 hours she was 0.5 miles past Grandma's House.That there is no time with the Distance to Grandma's House equal to zero indicates that the time step is to large. This simply tells us that our Time Step is too large for the relationships in the model.

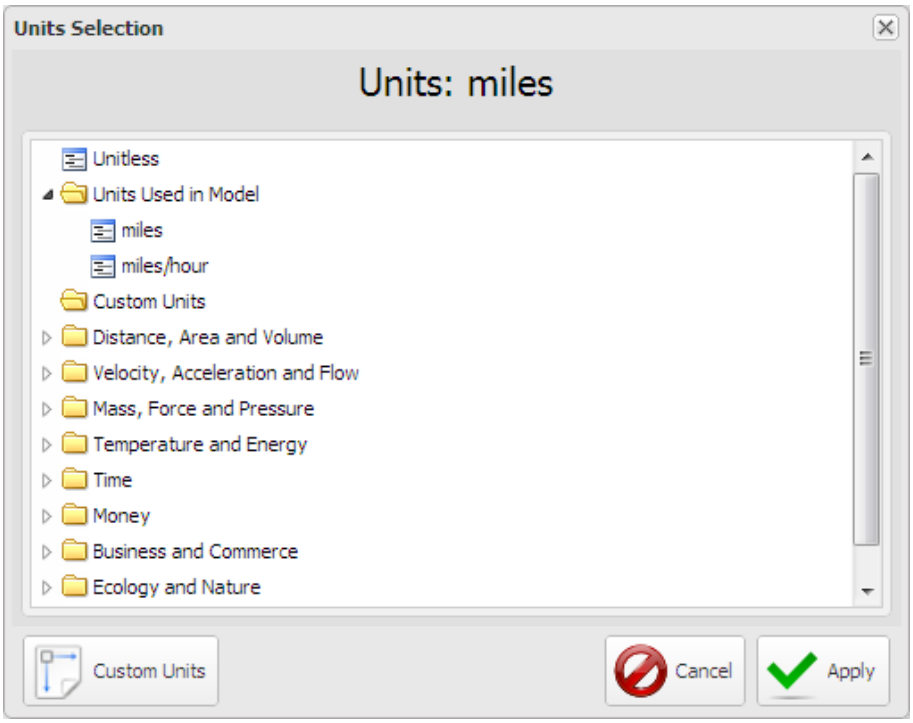


Figure 5. Units Selection Window

Exercise 3 - 1

Run this model of Figure 1 with a Time Step of 0.25 and 0.125 and from the Tabular Display which Time Step do you think is most appropriate and why?

Figure 8 represents a modification of the Little Red Riding Hood walking to Grandma's house which was presented in Figure 1.

Hopefully you found in Exercise 3-1 that both 0.25 and 0.125 produced a step with a distance to Grandma's House of 0 at 2.25 hours. In finding no difference between the results for 0.25 and 0.125 you should have concluded that 0.25 was a small enough for this model.

The time step in Figure 8 has been set to 0.25 and the Stop at Grandmas variable has been added to tell the simulation to stop when Red has reached Grandma's House. If you run the model you'll see that it does just what we want it to do though if you look at the formula in Stop at Grandmas some explanation is probably in order.

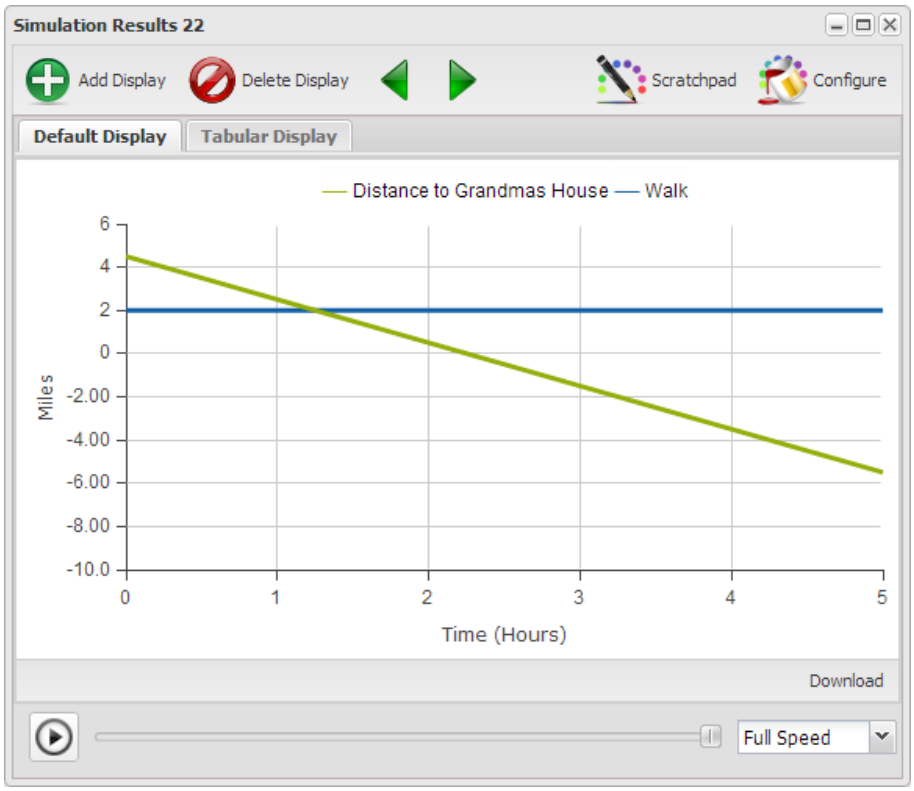


Figure 6. Walk to Grandma's/Graphic Results

```
IfThenElse([Distance to Grandmas House] < {0 miles}, STOP, 0)
```

If you look at the Configuration Panel for Stop at Grandmas you'll notice that the Units are Unitless. The variable itself doesn't need a definition of units because it's not participating in any calculations. It's just a test.

As for the formula what you should remember is that when you start using units in a model, which you should do, all formulas have to be consistent from a units perspective otherwise Insight Maker will raise an error message. Just as a test change {0 miles} to 0 and run the model. Because [Distance to Grandmas House] has units what it is compared to has to have units.

Exercise 3-2

In the [Stop at Grandmas] variable change {0 miles} to {0 kilometers}. Does the model still work? Why?

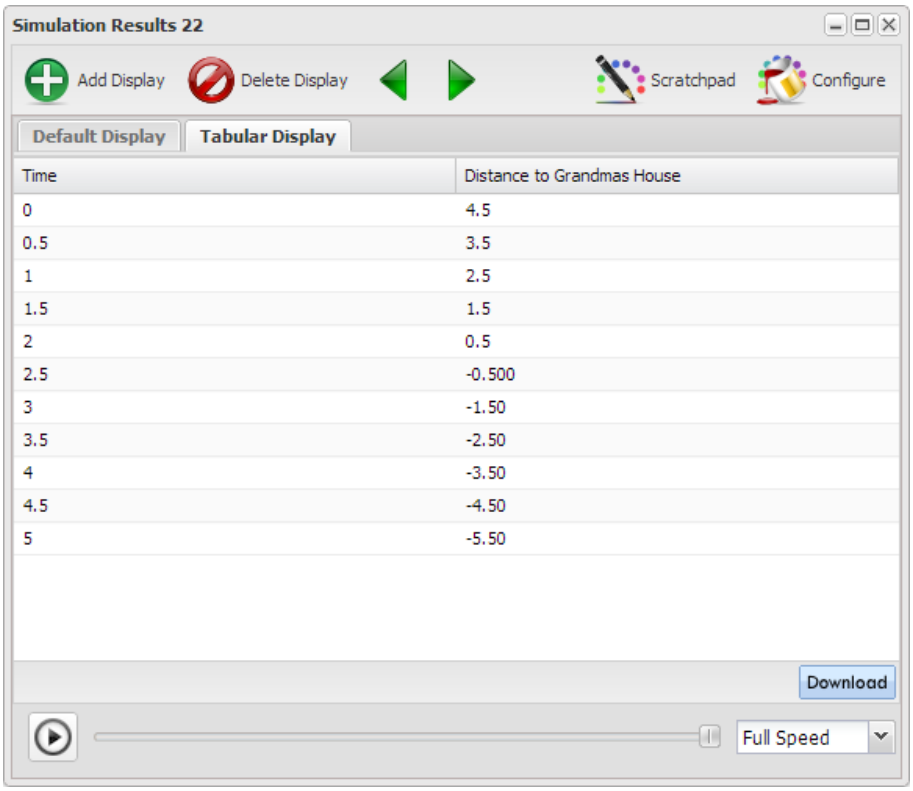


Figure 7. Walk to Grandma's/Tabular Results

The simulation engine in Insight Maker is smart enough to convert between the myriad of similar dimensions, e.g., miles, kilometers, feet, etc. Though it's recommended that you make conversions explicit otherwise models become very difficult to understand.

Exercise 3-3

Seldom is there ever just one right way to build a model. You build the model to help you understand something and you might do that in different ways. Even for a model as simple as Going to Grandma's can be structured in several different ways other than starting with a stock of 2.5 and reducing it by walking. Try to build one or two alternatives to this model.

Hopefully the Going to Grandma's model has given you a sense of an approach for developing models along with some useful tips and an introduction to using units and why they can be so useful to you. Oh, and not forget about putting notes in the model. Wiring diagrams without knowing what the pieces mean are generally not very useful.

Why Aren't We All Rich

The following is intended to be another example of he development of a model, though somewhat more involved than the previous one. Here's the initial question describing what I'd like to understand.

> If one can put money in an investment account and it grows over time, and it grows even faster with regular deposits, why aren't more people rich and ready for retirement? I've started numerous retirement programs through the years though for one reason or another they've all evaporated in time. What is the basis of this sad state of affairs?

Figure 9 presents the initial set up for this model.

- **Investment Account**. represents the amount of money, in Dollars, in the account. If you look at the Configuration Panel you'll notice that Units are set to Dollars.
- **Initial Deposit**. is a variable used to specify the amount of money that is initially put into the [Investment Account] when it is opened. Remember we said only a flow can increase or decrease a stock, though you can use a external variable to set the initial value for a stock. This is done done to make the [Initial Deposit] explicit with a slider for testing. The Units for [Initial Deposit] is also set to Dollars.
- **Time Setting**. We've assumed that this is an investment account that will compute and add interest on a monthly basis so the time settings are set up to run for 36 months with an initial Time Step = 1 knowing that we will have to test this later on.

If you run the model you'll find out it's about as interesting as watching pain dry, thought it does run.

Modeling Tips

Before you run a model you should develop a sense of the result you expect from the model at this point of its development. Then once you run the model you should be certain that is it performing as expected. When the result is not what you expect then either the structure is wrong, your assumptions are wrong, or you simply have an opportunity to develop your understanding.

You should never be more than a single concept change away from a running model that produces a result that you understand. You may think this a bit strict though after you add several elements to a model and it doesn't work and you spend hours trying to figure out why you may have a better appreciation for this statement.

Since this is an investment account that is supposed to grow based on monthly interest Figure 10 provides a few additions.

Annual Interest Rate, as depicted in Figure 11, is the rate that will be used to compute the interest on the account on a yearly basis. Not the a slider has been included with a .01 step size to make it easy to test different values. Units is 1/year as this is the per year interest rate.

Months Per Year, as depicted in Figure 12, is just the number of months per year, a fixed constant of 12, to be used to convert the Annual Interest Rate to a monthly interest rate. The Units for this variable are Months/Year.

Modeling Tips

Making all the elements of a model visible makes it much easier for others to understand it. This is why Months per Year and Initial Deposit were created as explicit variables rather than embedding the valued inside other elements.

And what's definitely worth repeating is that providing comments for all the elements of a model will also make it much easier for others to understand. All one need do is mouse over an element and click on the "i" that appears to read the comment.

Interest, ad depicted in Figure 13, contains the calculation for the Interest at each step of the simulation. The Units for interest are Dollars/Month which is derived from the formula.

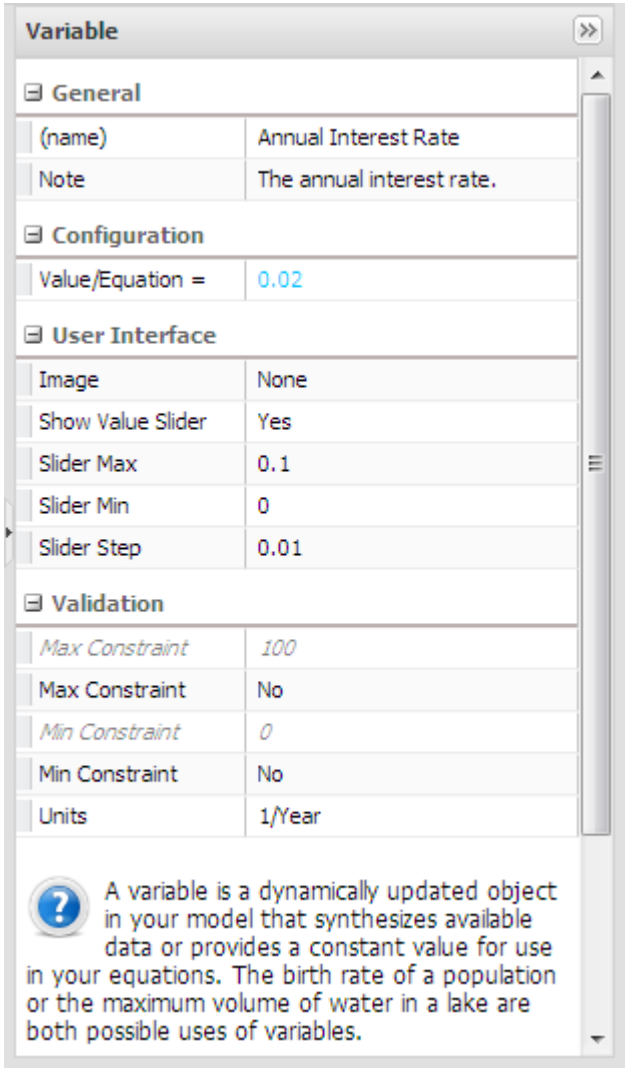


Figure 11. Annual Interest Rate Configuration Panel

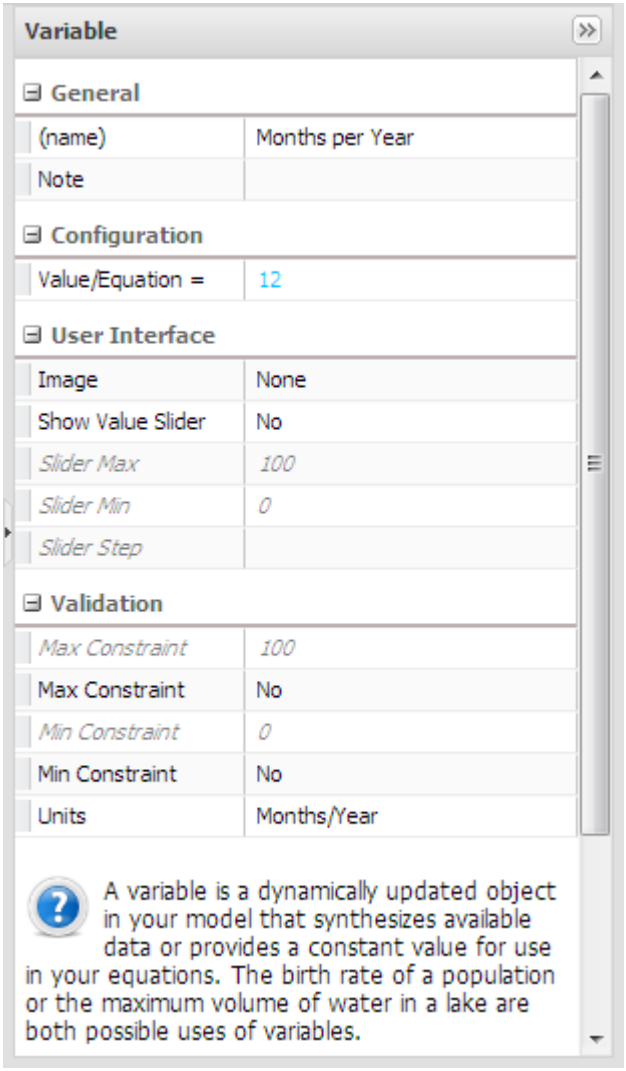


Figure 12. Months per Year Configuration Panel

[Investment Account] * ([Annual Interest Rate]/[Months per Year])

In Units: Dollars * (1/Year) / (Months/Year) = Dollars/Month

And as the simulation sums Dollars/Month over months the result added to the Investment Account is in Dollars which is consistent with the units specified for the Investment Account stock.

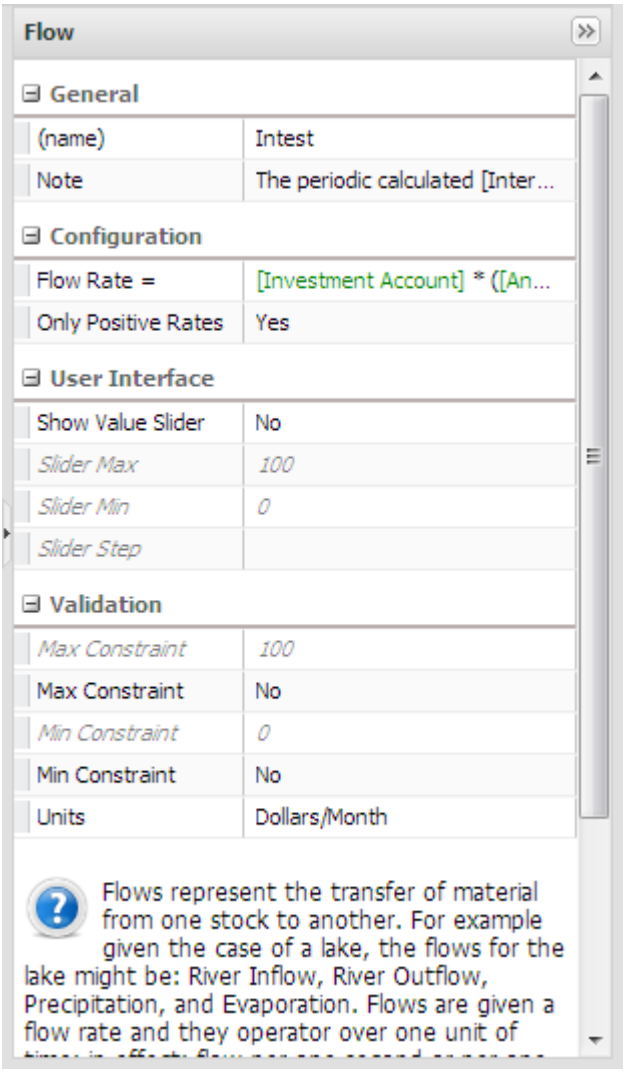


Figure 13. Interest Configuration Panel

Modeling Tip

R1 makes use of the Picture primitive used to indicate that the relationship between Investment Account and Interest created a Reinforcing structure, with the 1 simply meaning it's the first one in the model.

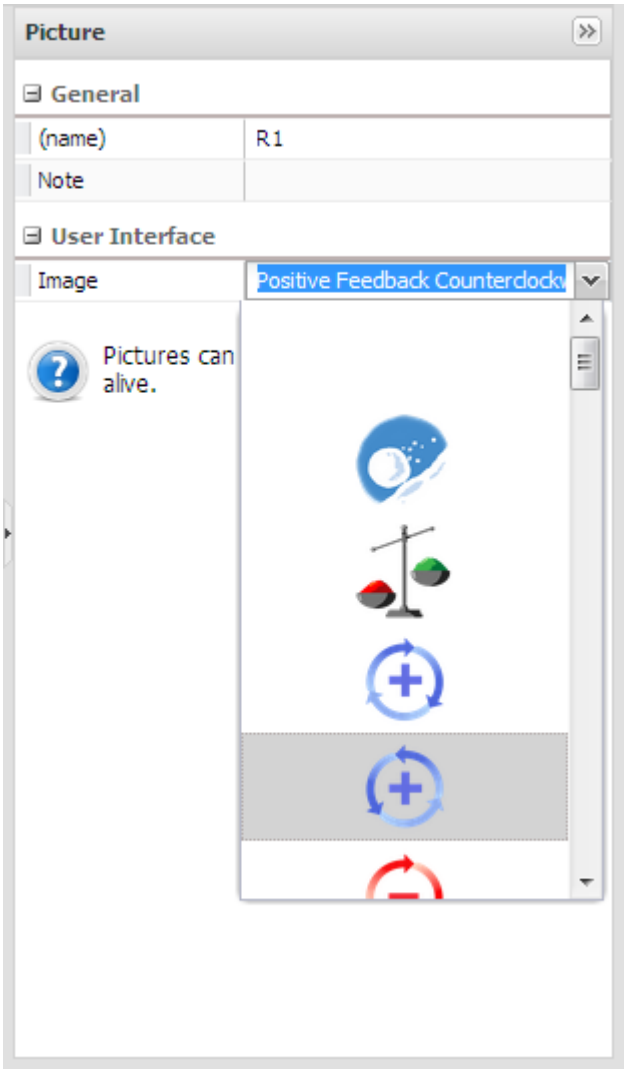


Figure 14. Picture Configuration Panel

You have the option of adding notes to the Picture element and there are a number of predefined images that you can select from the pull down that can be assigned to the element. There are images for balancing and reinforcing loops, both clockwise and counter clockwise. These pictures can be assigned to

Variables and Stocks also.

The other option is that you can put a URL in this field for an image somewhere on the web and that image will be displayed and may be resized.

Figure 15 depicts a run of this model over the three years with a 2% annual interest rate.

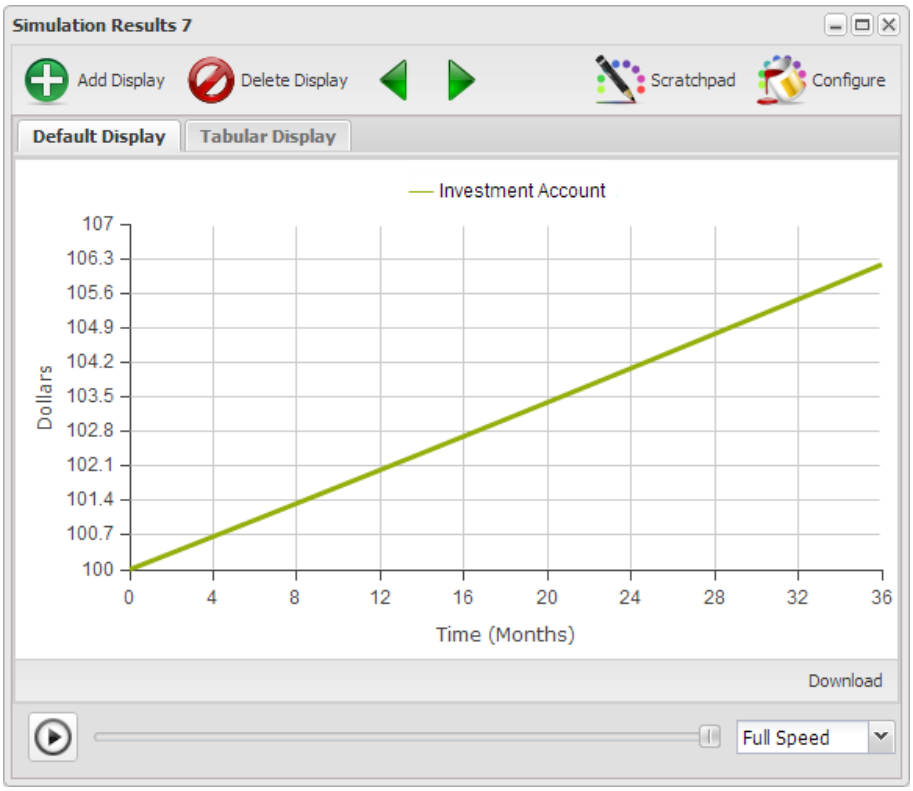


Figure 15. Why Aren't We All Rich v1

Admittedly $6 dollars in interest wouldn't seem like much of an incentive to invest in a investment account for three years. Though there are several aspects of the Investment Account that we might take into consideration.

Figure 16 depicts a couple several update to the model which are described below.

Time Settings have been changed so the model runs for 30 years, or 360 months.

Monthly Deposits have been added as a flow allowing one to indicate that there are additional monthly deposits into the investment account. A slider has been included to allow for testing this model with different monthly deposits. The Units for this flow are Dollars/Month, the same as for Interest.

Annual Interest Rate has been changed to 10% because one is likely to find an investment account that will average 10% over a period of 30 years, or so it would seem based on Whitfield & Co[1].

Figure 17 shows the new result from the model with these parameters. Though is this enough to retire on? Not likely.

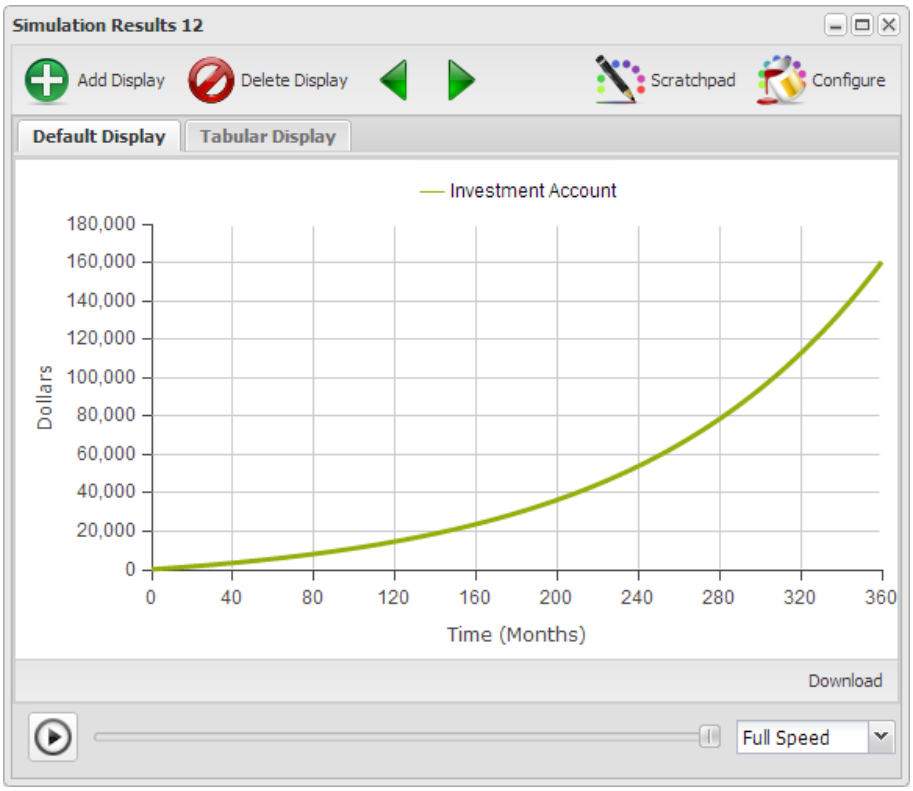


Figure 17. Why Aren't We All Rich v2 for 30 years, 10% interest, $70/month deposits

Figure 18 is the same model with the years changed to 40 years and with $100 Dollars/Month recurring deposits. And there's a significant difference between $160 thousand dollars and $640 thousand dollars. The difference being what you are willing to invest and for how long.

That said it's best if we don't lose sight of the initial question, that being why more people employ this model and become rich. Part of the difference between

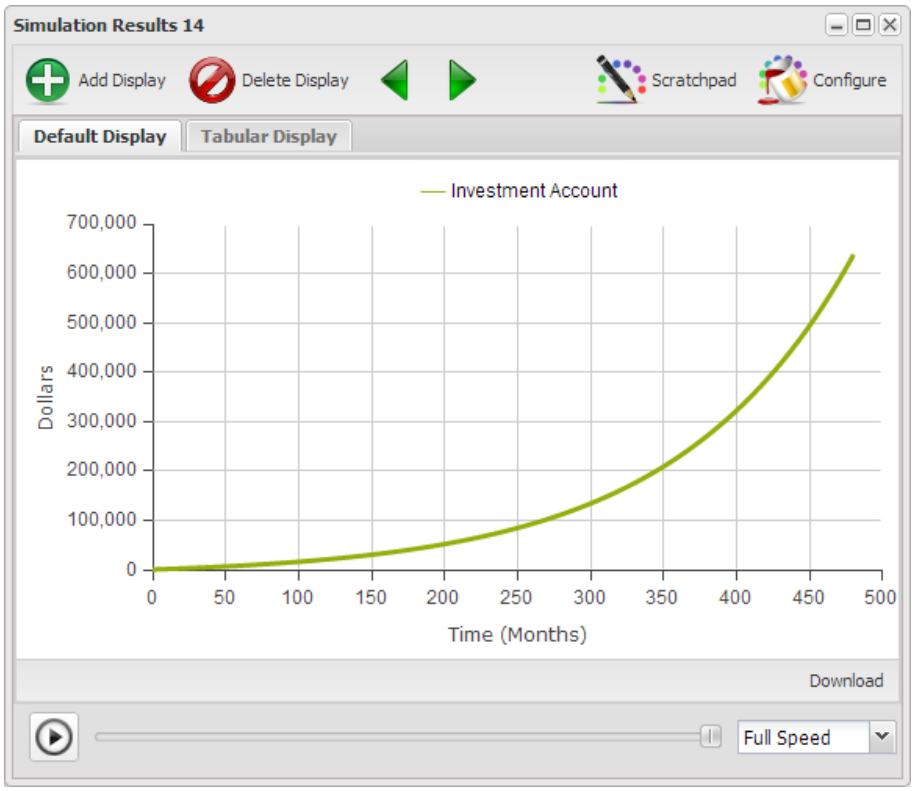


Figure 18. Why Aren't We All Rich v2 for 40 years, 10% interest, $100/month deposits

Figure 17 and Figure 18 is the extra $30 dollars/month in periodic deposits. One of the difficulties is finding the money to deposit on a monthly basis.

Figure 19 expands on where the money comes from for the monthly deposits.

Based on this model if one wants to increase the monthly deposits then it is necessary to increase Income or decrease Expenses as the Monthly Deposits are what's left over. Part of the difficulty is that when one has money the tendency seems for most to spend it rather than save it.

There are a couple additional aspects related to deposits that should be mentioned though won't actually be added to the model. Many companies allow employees to have payroll deductions directly deposited into a retirement account. This helps take care of the problem of having the money and spending it rather than depositing it in the investment account. Also, at least in the US their are tax laws that allow for the investment of some amount of pretax fund, money that you don't have to pay taxes on, to be placed in an investment account. The idea being that you would withdraw the money sometime in the

future when you're in a lower tax bracket. Some companies will even match a portion of your investment account deposits up to a certain amount each year. These options, which you could add to the model, would increase the resultant funds available at the end of the simulation.

The question we started with was that is this approach can be used to amass a sizable amount of money then why aren't more people using it to become well off. Part of the answer had to do with the idea that with money in their pocket people are more likely to spend it than save it even though there are incentives to save it.

Figure 19 provides an enhancement to the model adding Withdrawal and Penalty flows with some associated variables which are described below.

Penalty is levied by the Government if the funds are withdrawn before you reach 59 1/2 and is meant to be an encouragement to save. The % Penalty is a variable with a slider defined to you can test the value during runs. The Units for Penalty are Dollars/Month.

Withdrawal represents money taken out of the account to purchase things with. As the amount of money in the Investment Account grows it becomes more and more attractive for use to purchase other things and there develops a tug of war between the Attractiveness of the money in the Investment Account and one's Determination to Save. Attractiveness and Determination to Save both represented by percentages between 0 and 100%. Attractiveness is represented with a Converter, a modeling element not previously described.

Modeling Tip

It is often the case that a variable to be used in a model can not be represented as a constant or some well defined formula. The variable is actually a function of Time or some other variable. In the case of the model in Figure 19 Attractiveness is a function of Investment Account and is defined as a set of data points.

Figure 20 shows the Configuration Panel for Attractiveness Principle. Note that many of the configuration options are the same as other modeling elements. The ones that are different are in Configuration and Input/Output Table.

Because the variable is defines as a set of XY coordinates the Data has to be defined point by point as depicted in Figure 21, or the table may be imported.

Also notice on the Converter Configuration Panel there is an option for Interpolation. This option defines how Insight Maker figures out the Y values in between the defined X points. The graph displayed in Figure 21 depicts the Linear Interpolation meaning that Insight Maker treats the line between two points as a straight line and if computes the Y value from the XY values at the two points on either side of the X value.

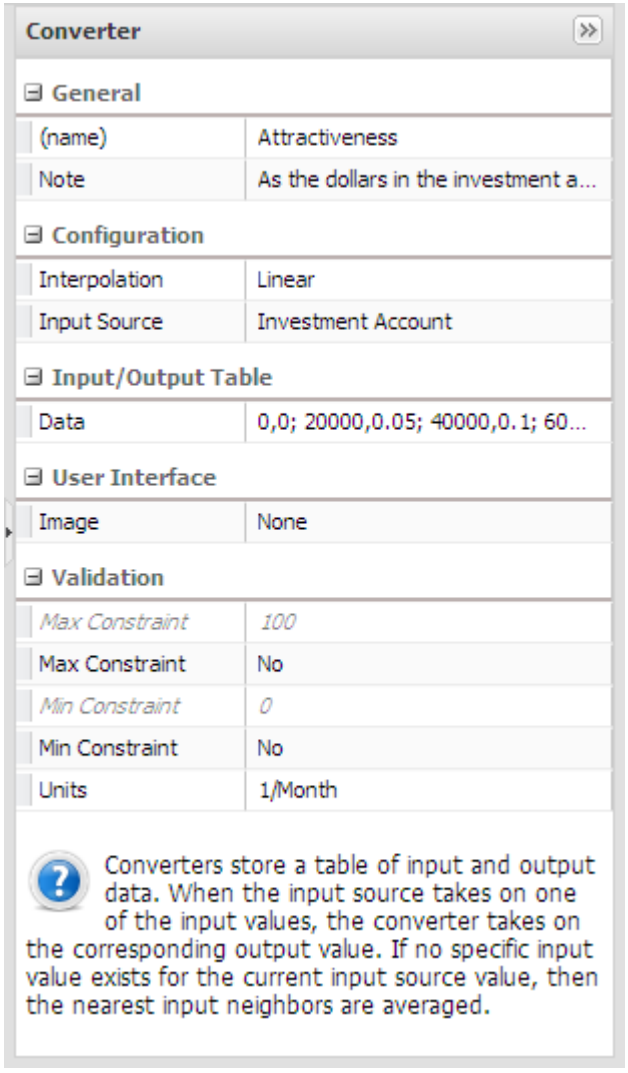


Figure 20. Attractiveness Configuration Panel

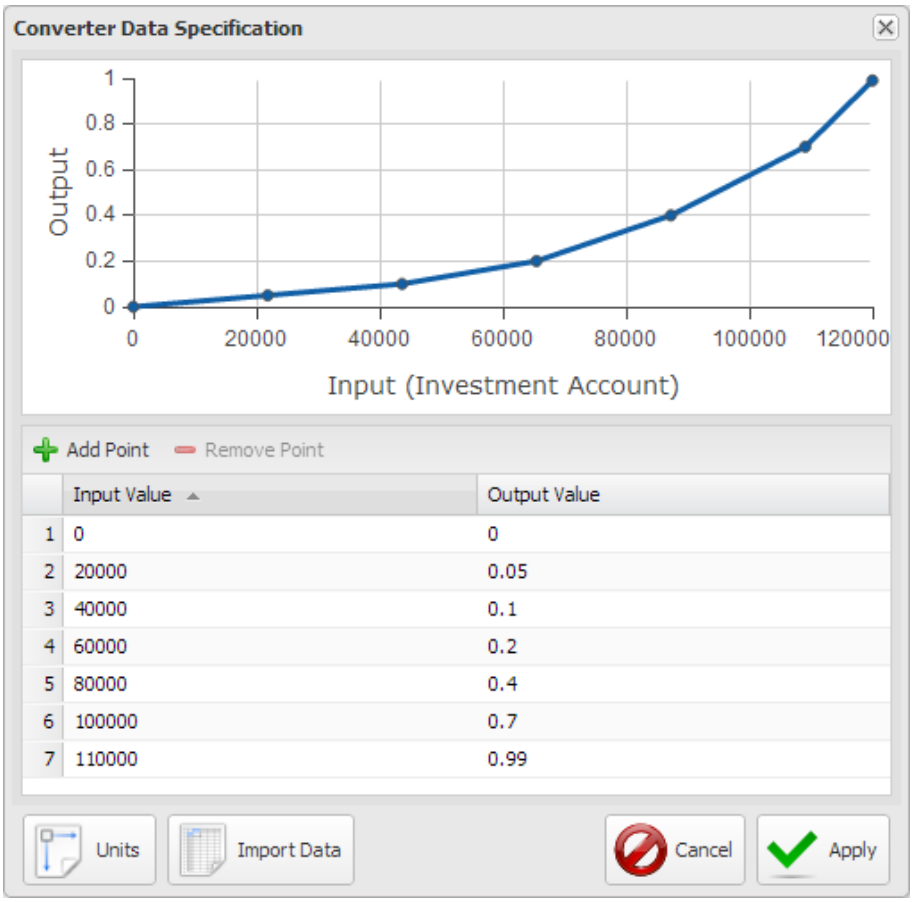


Figure 21. Attractiveness Data Specification

Figure 22 shows the curve for the Interpolation option of None meaning that it treats all the Y values between point X1Y1 and X2Y2 as Y1.

Figure 23, 24, 25 show the various display tabs for a run of this model with a Determination to save of 50%.

When the Investment Account reaches $87,000 dollars after 255 months it is sufficiently attractive to overcome the Determination to Save so money is withdrawn from the account every month and the account no longer grows. Is this a bad thing? That depends on the intent.

Figure 24 just shows that the Attractiveness has reached the Determination to Save level so withdrawals begin happening every month.

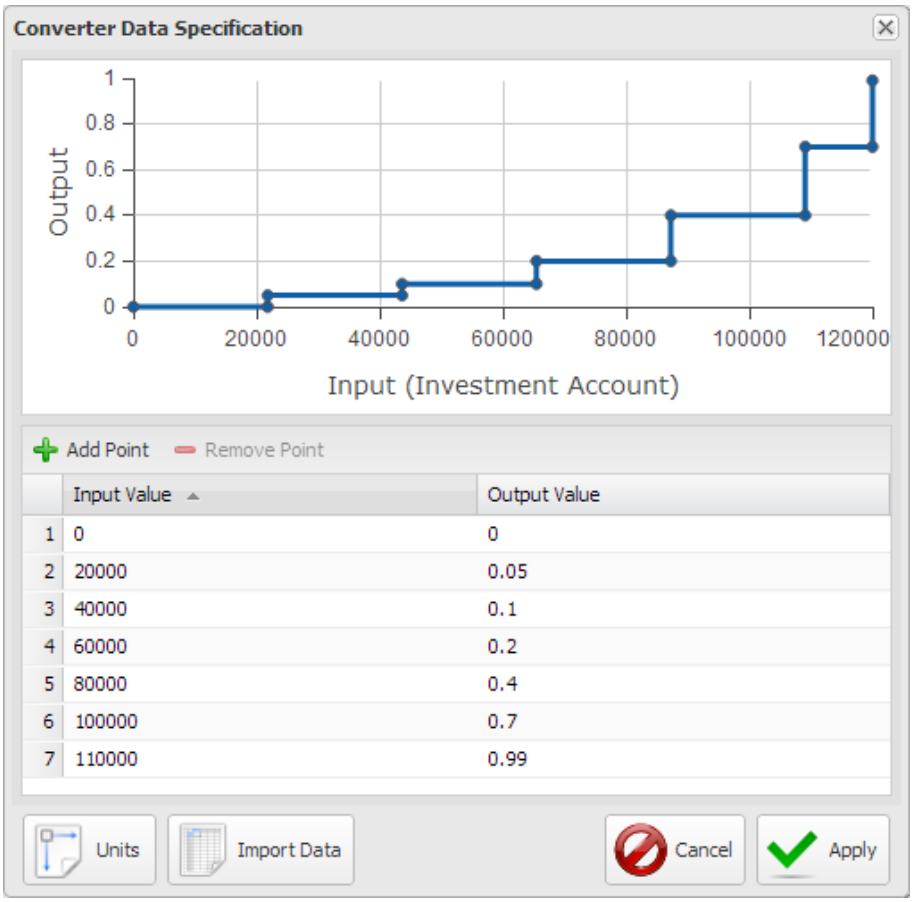


Figure 22. Attractiveness Data Specification with Interpolation = None

Figure 25 shows that there is almost $800 dollars a month being withdrawn from the account monthly and the account doesn't decrease. Maybe it's accomplishing what it needs to if $800 a month is sufficient to augment other income.

Note the large overshoot on the Withdrawal curve and a small one on the Penalty curve. This is most likely because the Time Step is too large. Figure 26 is the same display tab for the model run with a Time Step of 0.5. Notice how the curve cleans up.

Exercise 3-4

There is a logic flaw in this model which you might try to repair. The Penalty is not actually taken from the Investment Account but from the Withdrawal itself

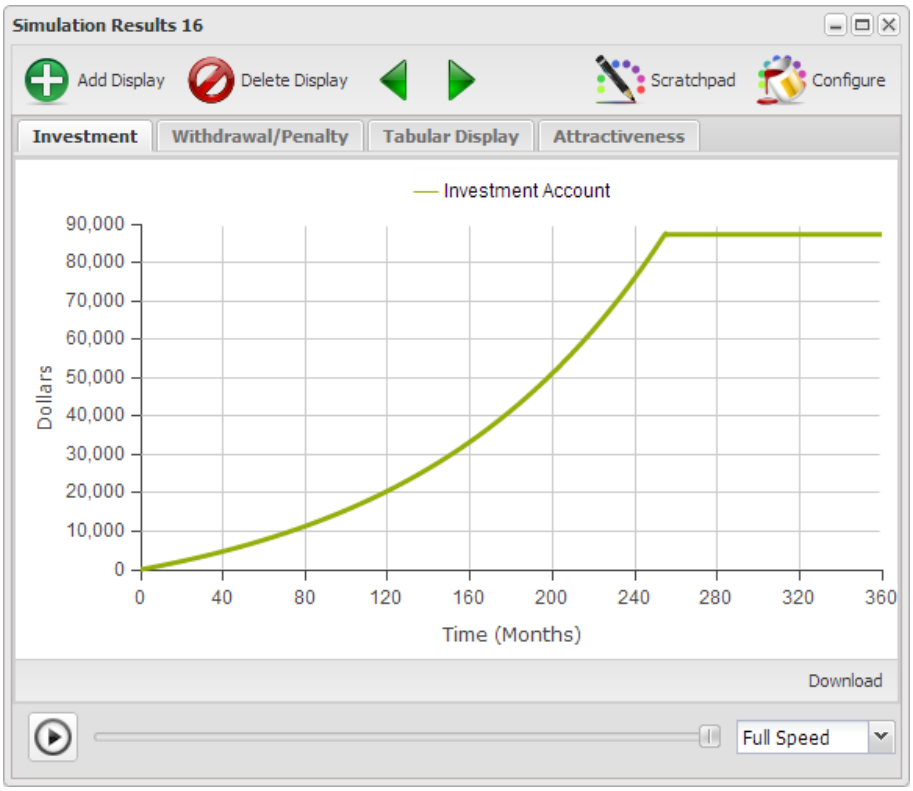


Figure 23. Investment Account Limited by Attractiveness

so it reduces the amount you actually get from the Withdrawal. Be warned that is might be a tricky fix.

We now have a model which provides some incentives to start and continue to deposit in an Investment Account, and some disincentives toward the withdrawal of funds, though have we really addressed the initial situation posed? Not really. As far as starting the Investment Account and regularly depositing money, there are incentives, and for many these incentives were enough to get them to invest. For many the incentive, for one reason or another, has not been sufficient. And, any more strict incentives would likely be looked on unfavorably. People do not like to be manipulated, even when it is for their own benefit. The penalty for withdrawal is a deterrent in some respects though as the Investment Account continues to grow its attractiveness in terms of what it can purchase continues to entice. The best answer for this situation is to legally tie up the withdrawal process so it's only an option in the case of dire emergencies. Though as much

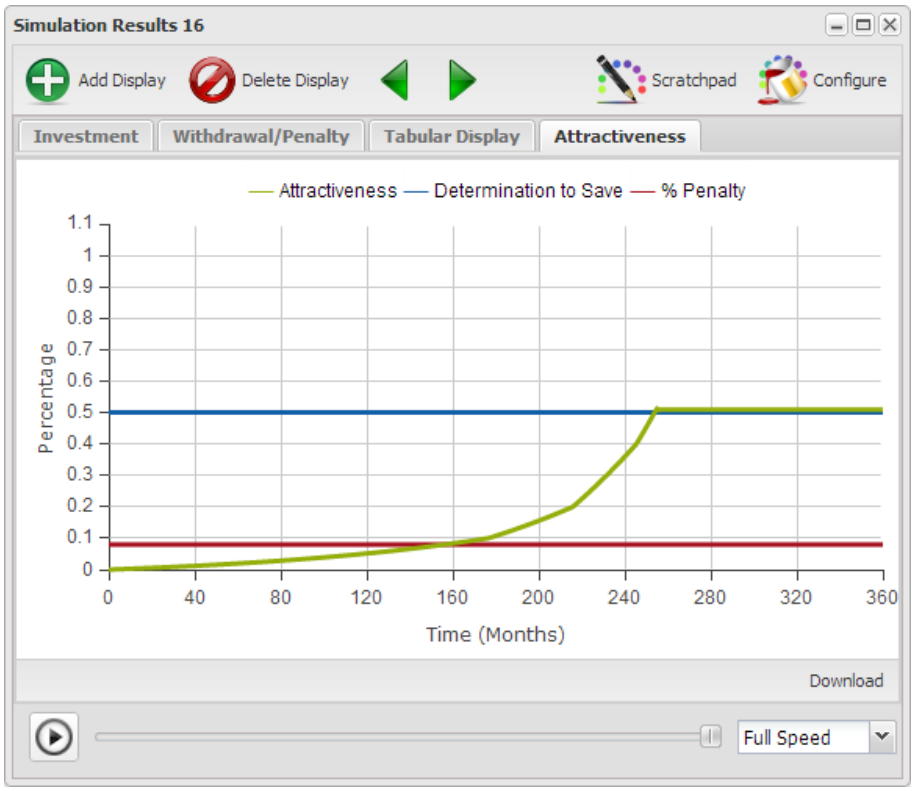


Figure 24. Investment Account Attractiveness and Determination

as people find being manipulated by others distasteful, being controlled by themselves is just as distasteful.

Is the model done? As usual, the answer is; "It Depends!" If it has provided sufficient understanding to address the situation posed then it is sufficient. If not then it should be taken further, though once it is sufficient you should STOP!

Building a Model Summary

- **Intent**. Be sure you have a good idea of what you want the model to help you understand. This may evolve as you develop the model.
- **Time Frame**. Ensure you have a sense of the time frame over which you intend to simulation the model. As you build the mode you may find you need to adjust your initial thought on this.
- **Stocks & Flows**. Identify the Stocks & Flows first as they are key elements of the model.

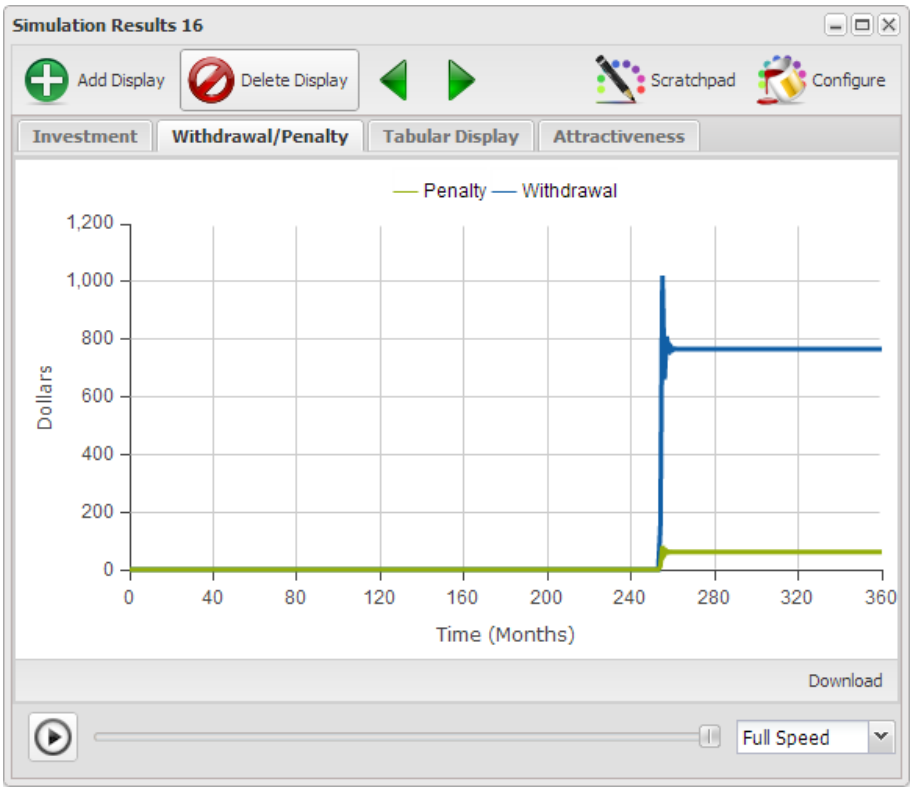


Figure 25. Investment Account Withdrawal and Penalty

- **Use Units**. Units help to ensure your model is sound and Insight Maker will test for consistency of units. If the units are consistent it doesn't guarantee the model is sound though it does add a level of confidence.
- **Variables & Links**. Add Variables & Links to influence the flows.
- **Test Often**. Each time you make a logical addition to the model think about how you expect the model to behave then run the model and see if there is agreement with your expectation. If it isn't then it's an opportunity to learn and improve the model. And if it does agree you should still consider the output. It may be that your expectation and the model are both wrong.
- **Time Step**. Test the Time Step to ensure it's small enough to capture all relevant transitions in the model.
- **Stop at the End**. When the model serves the purpose for which you are developing it, STOP! There is always more you can add to a model. You should only include what is relevant to satisfy the initial intent.

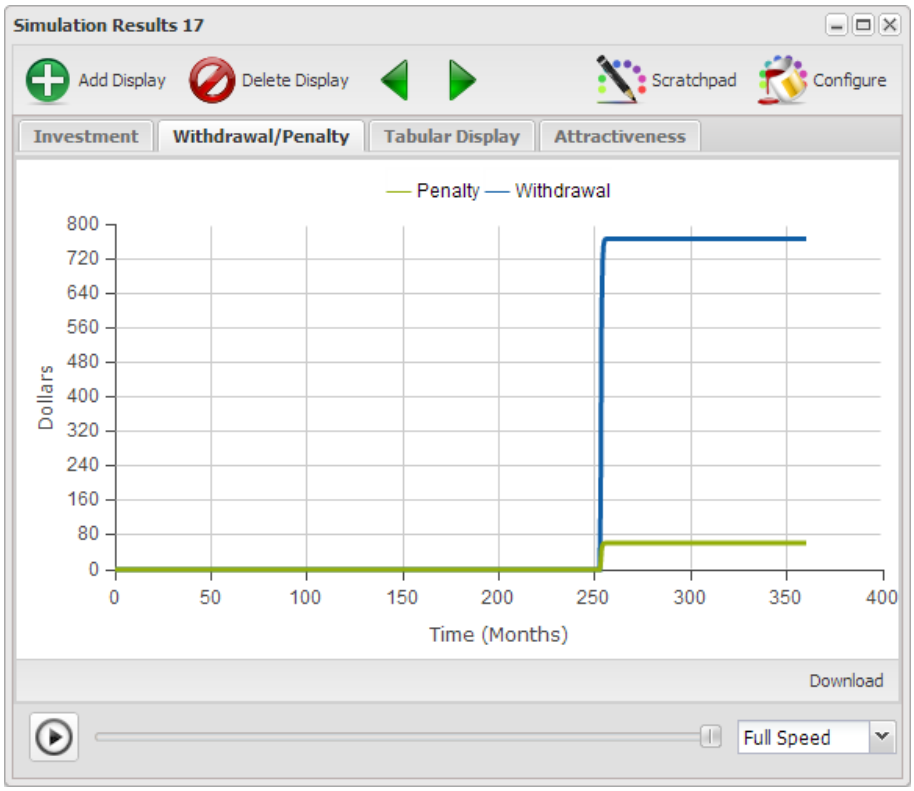


Figure 25. Investment Account Withdrawal and Penalty with Time Step = 0.5

References

- Catalina Foothills School District. 2003. Tips for Using System Dynamics Tools. http://www.clexchange.org/ftp/documents/Implementation/IM2003-12TipsUsingSDTools.pdf
- Ford, Andrew. 2009. Modeling the Environment. http://www.amazon.com/Modeling-Environment-Second-Andrew-Ford/dp/1597264733/
- Keeting, Elizabeth K. Internet 2013. Everything You Ever Wanted to Know about How to Develop A System Dynamics Model, But Were Afraid to Ask. http://www.systemdynamics.org/conferences/1998/PROCEED/00024.PDF
- Newell, Barry & Proust, Katrina. 2012. Introduction to Collaborative Conceptual Modelling. https://digitalcollections.anu.edu.au/bitstream/1885/9386/3/Newell_Introducti
- Whiftield, Caraig. 2012. What Returns Should We Expect from the Stock Market. http://www.whitfieldco.com/blog/?p=39

Chapter 6

Chapter 5 Implications of Reality v1 13.05.xx

Notes to Reviewers

This chapter is actually a series of mini chapters collectively with one intent. That being to demonstrate how the same model can be applicable across apparently unrelated areas of study. This will be accomplished though the development of patterns of behavior which occur in every area of study imaginable and then showing how that pattern can be represented by a relatively simple combination of reinforcing and goal seeing structures that you learning about in Chapter 2.

The following chapters actually represent the most common systems archetypes though we're not going to surface that fact until later. And we're definitely not going to use the term Systems Archetypes. The intent is to simply, via examples, extend the two basic structures and produce models for each of these extension.

What we expect will happen is that the reader will develop an understanding of the value of critically thinking about the possible implications of the current structure and the various ways in which it may evolve into more complex structures.

For each chapter there are real world examples which are then developed into structure. The reader is asked to interact with the models provided and then asked to identify similar structures from their experience, clone the existing structure and update the model to reflect their example. And once the model is created it should be simulated.

Three Basic Structures

In the previous chapter we covered the three basic structures of Figure 1 in some detail. And it was claimed that all the models you will ever create will simply be a combination of some number of these basic structures. We don't expect that you take this on faith and while we can't prove it in this chapter we will provide you an opportunity to experience some of the more common structures which occur repeatedly across all disciplines of science.

Growth Slows Over Time

Figure 2 presents a structure composed of a growth structure, R1, with a balancing structure added on the right. The only difference in the balancing structure is that it is set up in such a way that it doesn't have an effect on the results until the results reach a certain level.

- Rabbits tend to multiply very rapidly so why is it we're not completely overrun by rabbits, well maybe everywhere except Australia?
- – keep playing instead of cleaning up the mess in the room, which makes further play difficult, AND the increased mess repels the kid from cleaning up.

Limited Resources Are Shared By Others

Growth Leads To Decline Elsewhere

Partners For Growth Become Adversaries

- Parents can be interesting. Both Mums and Dads want to act as good role models and create a positive family environment, but sometimes, each one of them gets caught up in being seen as ''the good parent" and gives in to our whims and desires. We all know how this works: If one parent says 'no' to something, we just ask the other parent. The problem is that when one or both of our parents wants to be seen as the ''good parent,''the other one ends up being seen as the 'bad' or 'tough' parent, and the whole 'good role model' or 'good family environment' disintegrates pretty quickly. https://dl.dropboxusercontent.com/u/102156844/Loop%20Structures/Family%20dysfunctionality%

In Time The Problem Returns

- Your soccer ball is soft so you put air in it though in a few hours you have to put more air in it. And after a few weeks it seems like you spend all your time pumping up your soccer ball.

Often times what appears to be the most appropriate way to deal with the situation doesn't really solve the problem and in time actually makes the situation worse.

The Fix Creates A Problem Elsewhere

– a kid telling a small lie to cover a small thing, only to find himself into further problems that would require bigger lies, making it all the more difficult to tell the truth later when two people walking against each other on the street try to pass by making a step sideways only to discover they block each other again and another attempt to go off the way leads to the same situation

The Fix Overshoots the Goal

Dependence On The Fix Develops

- in organizations, people firefighting instead of solving the root problems, thereby reducing time available to invest in solving root problems
- I'm sure we could easily set up a similar example for short term economical measures that redirect resources from what could fix the economy in the longer term, though any model would probably be controversial.
- – Wanting to win all the time can be really hard, and to increase your chances of winning, there are two options. One option is to cheat. The other option is to practise all the time and demonstrate a serious commitment to your sport. Whilst cheating is probably easier, it becomes highly addictive (because it doesn't require much effort) and because of this, the desire to win fairly though practise and commitment to your sport doesn't happen. Think of what happened to Lance Armstrong. https://dl.dropboxusercontent.com/u/102156844/Loop%20Structures/Lance%20Armstrong%20Shifting%20

Everything is Connected

The chapter should explicitly depict the relationships between the structures presented in the previous sections and explain the natural evolution paths for the structures.

** Figure x. Systems Archetypes Relationships

Chapter 6 - Applied Understanding

Chapter 7

Models and Truth

All models are wrong, but some are useful – George E.P. Box

A model is a tool designed to reflect reality. A model is never a perfect mirror of reality, but often models can still be useful even given their imperfections. In this chapter, we will take a journey exploring different types of models and distinctions that are commonly used to classify and understand them. We will consider several approaches to modeling that are quite different from the ones we have introduced throughout this book. These will help to understand the richer ecosystem of modeling tools and techniques that exist and how the ones we have learned fit within this ecosystem.

The ultimate destination of this exploration will be a clear understanding of the fundamental principles and approaches used to construct models. There will be many detours that we must make to arrive at this destination, but in the end we will be able to divide models into two overarching categories based on their purposes and the techniques used to construct them. By mastery this divide, and how the work we and others do fits into it, we will obtain a rich perspective and understanding of the relationship between models and truth. We will also have a renewed appreciation for the strength and power of the techniques introduced in this book for tackling a wide swath of modeling problems.

Before we get there, however, let's introduce some of the terminology that is commonly used to describe models. It is useful to take a step back and first discuss different kinds of models. Modeling is a wide-ranging field with many distinctions made by modelers and mathematicians. Three of these distinctions are presented below:

Deterministic versus Stochastic Models

There are two polar opposite views of the world. One view says the fate of the universe is governed by strictly predictable laws of physics. In this view, the

universe acts as if it were a giant machine, where if its current state is known (down to each individual atomic particle), its future states through the rest of time are predetermined. The opposite view is that the universe is ruled by chance and randomness. Random quantum mechanical fluctuations merge and amplify leading to an infinite range of diverging possibilities.

Which of these two views holds more of the truth? We certainly do not know and it is possible that this will be a question that physicists will never cease exploring. Albert Einstein had a viewpoint of special interest, however. He was a strong partisan of the more deterministic view, famously remarking, "God does not play dice with the world."

When creating a model of a system, we must choose how we treat chance. Do we build our model deterministically, such that each time we run it we obtain the same results? Or do we instead incorporate elements of uncertainty so that each time the model is run we may see a different trajectory of outcomes?

Mechanistic versus Statistical Models[1]

When beginning to build a model of a system, there are many questions you should ask, two of which are:

1. Do I know (or have a hypothesis of) the mechanisms that drive the system?
2. Do I have data that describe the observed behavior of the system?

If the first question is answered in the affirmative, you can build a mechanistic model that replicates your understanding (or hypothesis of) the true mechanisms in the system. If, on the other hand, the second question is answered in the affirmative, you can use statistical algorithms such as regressions to create a model of the system based purely on the data.

If neither question is answered affirmatively... well in that case there isn't much of anything you can build.

Aggregated versus Disaggregated

When building a model, the issue of scale becomes very important. Imagine we are concerned about the effects of Global Climate Change on water resources.

[1]This relates, more broadly, to the contrasting research approaches of induction and deduction. Induction starts with data and observations which are analyzed to create a broader theory (similar to a statistical approach to modeling). Deduction starts with a theory and finishes with the collection of data to confirm the theory (similar to a more mechanistic approach to modeling). It is easy to become mixed up with the meanings of induction and deduction and even great minds have confused them. While Sir Arthur Conan Doyle's character Sherlock Holmes attributes his impressive powers to "deduction", he is actually using induction. Treating what we are calling "statistical" models here as a form of induction, we can also refer to them as "phenomenological" or "empirical" models.

We may wish to examine the question of whether there will be sufficient water supplies given a rise in future temperatures.

At what scale do we build this model? The range of possible scales is wide:

- At the most aggregate, we could estimate total worldwide water demands and supplies into the future.
- Maybe that is too coarse a scale, however, as clearly having excess water in Norway has little direct impact on the situation in Egypt. We could instead create a finer resolution model that separately looked at water demand and consumption in each country.
- Even that may still be too coarse, maybe we should make our model more granular to look at specific cities or population clusters around the globe.
- At the extreme disaggregated level, we might even want to model individual people – all 7 billion of them – and their needs and movements around the world.

There is no simple answer to this question of optimal scale. The best choice is highly context-sensitive and depends on the needs of the specific modeler and application.

Prediction, Inference and Narrative

The three distinctions just presented – deterministic vs. stochastic, mechanistic vs. statistical, aggregated vs. disaggregated – can be used to classify models. We can even use them to classify the models we have discussed in this book. Most of our models would be classified as deterministic (random chance is generally not explicitly incorporated in these models), mechanistic (we generally assume mechanisms rather than estimating dependencies from data), and highly aggregated (the agent based models are an exception).

There are many nuances to these broad distinctions that can also be made (e.g. the type of statistical techniques used for a statistical model) and there are also many other distinctions that can be made between model implementations such as the programming language or software that was used to implement the model. These distinctions and technical choices are important when constructing a model, however what is of key importance is the utility of the model for fulfilling a specific goal.

Technical details matter – they can affect maintainability and other factors – but they are of secondary interest to the adequacy of a model in fulfilling it main purpose. It would make as little sense to say a model was fundamentally bad because it was written in a relatively ancient programming language – like Pascal – as it would be to say a model was fundamentally bad because it was, for instance, deterministic. Let's look back at Box's quote at the beginning of

this chapter. We know all models are wrong, what we should really care about is their utility in meeting a specific task.

So instead of using the aforementioned technical classifications to discuss models, we think it is more useful to base our discussions of models on the model's driving purpose. This allows us to leave behind relatively mundane technical and implementation details to focus on what we really care about. Among the many different reasons for building models, they all boil down basically to three broad purposes: prediction, inference and narrative.

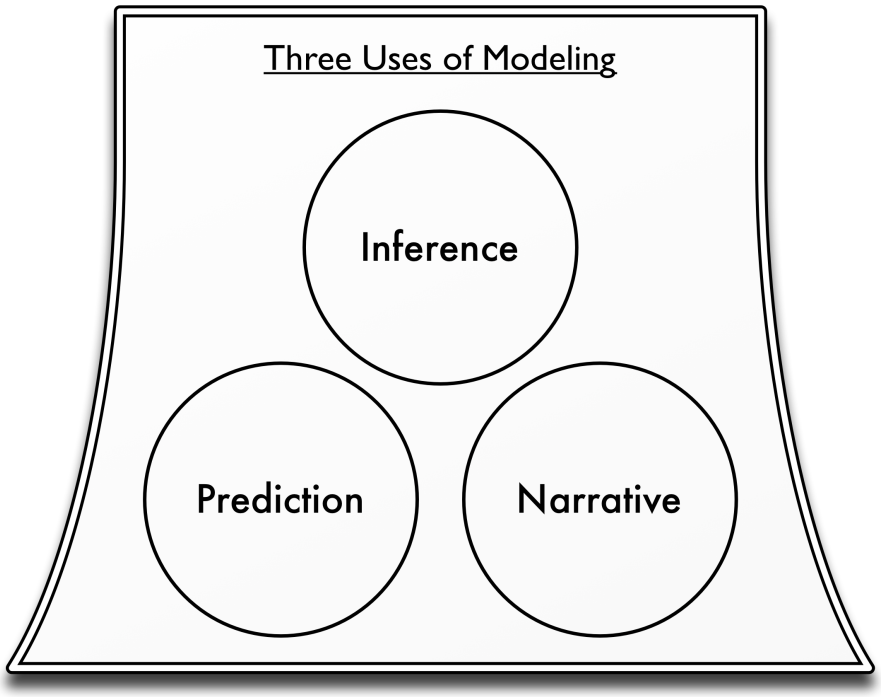


Figure 1. Three Usages of Models

Prediction : Models used for prediction are the most straightforward. They attempt to forecast some outcome given information about variables that may influence that outcome. A weather forecast is an example of a model used for prediction. Likewise, when you apply for a credit card at a bank, they run a predictive model to determine your risk of not paying them back what you owe and defaulting. When you apply for life insurance, the company has a model

that predicts how long they think you will live in order to determine how much they should charge you. All these models take in data (the current temperature for the weather forecast, the amount of money in your bank account for your risk of default, your age for the life insurance application) and apply various forms of analysis to generate a prediction of the outcome.

Inference : Models used for inference are most common in academic research. Often, academic research questions distills down to this simple template: "Does X affect Y?" These are inferential questions. As an example, a researcher may make a hypothesis statement such as, "The wealthier a high-school student's family is, then the higher the student's test scores will be." The researcher may then build a model to test the validity of this hypothesis and the model's results will generally be phrased in terms of a p value indicating the statistical significance of the evidence in support of the hypothesis.

Narrative : Models are often used to tell a persuasive story. When the Obama administration wanted to persuade lawmakers and the public to support their economic stimulus, they famously published the graph shown in Figure 2. A great deal of complex modeling and mathematics surely went into constructing this figure. However its core purpose was to tell the nation a story: Things are going to be bad, but the recovery plan will make them less so. Such stories are at the heart of narrative models and we will return to this figure later on and why it is not really a predictive model despite it generating predictions.

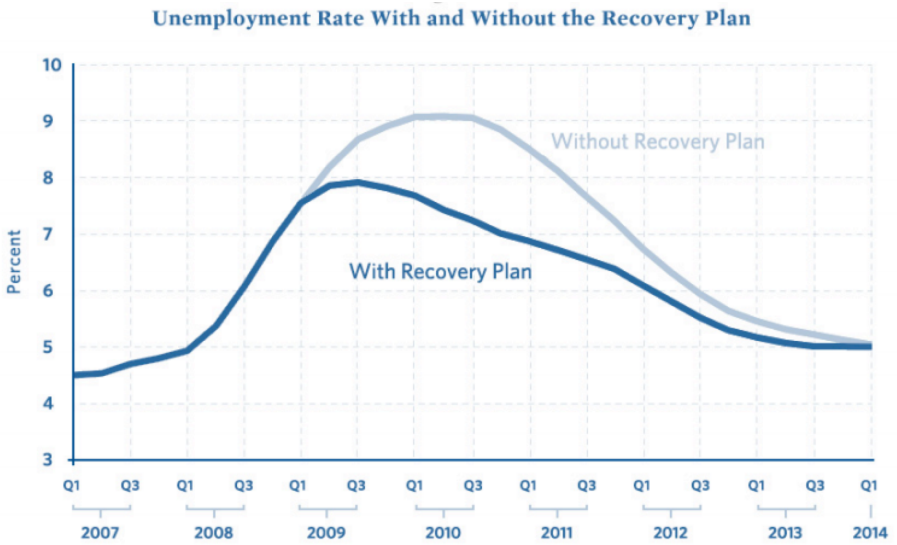


Figure 2. The Obama Administration's Predictions for the Effects of the Recovery Plan [@Romer:2009tx]

All models can be classified in terms of these three primary purposes and we

will see how useful it is to discuss modeling projects in terms of them[2].

The Strange Case of Inference

To help us get at this fundamental classification scheme, let's first talk for a moment about the process of inference. Take the earlier example of determining whether wealth results in increased high-school test scores. We phrased this hypothesis in a specific way: that increased wealth will always increase test scores. This illustrative statement, however, actually differs from what is often done in practice. In general, researchers simply asks the question "Does X affect Y?" rather than "Does X increase Y?" It's just a slight difference, but it is a more flexible question that allows for many forms of relationships. For our example, we would ask the question "Does wealth affect tests scores?"

The gold standard to answering questions like this is the controlled experiment. Controlled experiences allow you to develop strong inferences as you can see how a system responds when you hold all variables constant except for the single one you are interested in. For our example, we could imagine an experiment where we took a sample of a thousand families from a school district. When these families' children enter high school we would randomly select them to be in a "poor" category and the other half to be in a "rich" category. Families in the rich category are given grants of $500,000 a year to spend how they wish while the parents in the poor category are fired from their jobs and have their savings frozen for the duration of the experiment. Once the students graduate from high school, we would compare the scores for the students in the poor and rich categories.

These controlled randomized experiments are considered the ideal approach to answering inferential questions like these as they allow you to truly determine the effect of your variables, in this case wealth. For many types of questions, such experiments can be implemented (for instance does consumption of a new drug help treat a disease). Unfortunately, in general complex social questions are simply impossible to answer with them. We can consider the testing procedure we just imagined to assess the effect of wealth on scores, but it would be impossible and unethical to undertake in a real community. Furthermore, even if you were to implement the experiment as described, the behavior of a family that was poor or wealthy to begin with might very well differ from a family that experiences a sudden change in income.

[2] And we strongly recommend doing so. It is important to clearly define the purpose at the start of a project. The techniques used and data required depend significantly on the model's overall purpose. To be very clear, it is important to clarify at the outset whether your primary goal is to use a model for prediction or for narrative. Many modeling projects may attempt to do both only to find themselves with a model that does neither.

Traditional Model Based Inference

Given our general inability to undertake ideal controlled experiments, how do we answer inferential questions? The standard way is to collect data and then construct a model enabling us to measure the statistical significance of our hypothesis given the data. Due to history and simplicity, linear regression models are by far the most commonly used type of model today. A linear regression predicts an outcome (Y) based on the multiplication of variables (X's) by a set of coefficients determining the effect of the variables on the outcome (β's):

$$Y = \beta_0 + \beta_1 \times X_1 + \beta_2 \times X_2 ...$$

For the education example we could collect data on a number of students, measuring their families' wealth (X_1 in the equation above) and the student's test scores (Y). We would then run the linear regression to determine the coefficient values (β_0 – the intercept – and β_1 – the effect of wealth on test scores). If we thought there were other factors that affected test scores, we could measure them and include them as addition X's in the regression.

In addition to obtaining the values of these coefficients, we also obtain as a result from the regression the statistical significances or "p values" of these coefficients. Although p values are commonly used in statistics, they are ubiquitously misunderstood[3] so it is useful to briefly review them.

In short a p value measures the probability of seeing the measured data (or more extreme data) assuming the null hypothesis is true. Generally the null hypothesis will be that there is no relationship between the variables and the outcomes.

When assessing the significance of coefficients, a p value means the probability of seeing that value of a coefficient (or one even further from 0), assuming that the (unknown) truth is that the coefficient actually has a value of 0. In other words, it is the probability of seeing the observed non-zero value, assuming that the true value is in fact 0. Frequently, probabilities of 10%, 5% or 1% or smaller are taken as indicating statistical significance. These low values indicate that the coefficient value is so far from 0, and the probability of this occurring by chance so small, that we can reject the null hypothesis and accept the fact that the coefficient is not 0.

Using the p values enables inference by relying on the statistical significance of the coefficients. If the probability of β_1 (the coefficient for the effect of wealth) occurring due to chance (given it is 0 in reality) is less than, say 5%, we can claim with reasonable strength that wealth does in fact affect test scores. This

[3]These misunderstandings are not only made by on-the-ground practitioners and analysts, they are frequently shared, and propagated, by university-level statistics instructors; see, for instance, @Haller:2002vo.

is the standard approach researchers take to model-based inference and is used ubiquitously.

A Troubled Sea of Assumptions

Let's stop for a second and consider what we have done here. In carrying out these logical steps to apply model based inference to determine whether wealth affects test scores, we have had to make one very large assumption: that the relationship between test scores and wealth is linear.

Our linear regression equation assumes that for every increase in one unit of wealth (X_1), test scores (Y) will increase on average by the amount of the coefficient (β_1). What if this were not in fact the truth? For instance, we could easily imagine the case where wealth initially helped test scores by providing students more resources and opportunities to learn. After a certain point, however, wealth might negatively impact scores as very wealthy students might lack the pressure or motivation to study hard.

If we believed this were the case, then our linear regression model from earlier would be wrong as would the inferences we obtained from the model. We could correct our model and inferences by changing our regression formula to contain a squared term that could replicate this type of relationship:

$$Score = \beta_0 + \beta_1 \times \text{Wealth} + \beta_2 \times \text{Wealth}^2$$

Using this equation, at low values of wealth the $\beta_1 \times$ Wealth term will have the most effect on scores. Conversely, at high levels of wealth, the $\beta_2 \times \text{Wealth}^2$ term will have the most effect on scores. Thus by having a positive β_1 and a negative β_2 we can model wealth as having an initially beneficial and then detrimental effect. If our assumptions about the quadratic relationship are correct, then this model will yield accurate inferences. If they are wrong, our inferences will be wrong again.

What are we really doing when we assume regression forms like this? Now it might not be immediately obvious, but what we are in fact doing is telling a story. Using our first equation, we are telling the story that as wealth increases test scores will almost always increase. Bill Gate's children will preform amazingly well here! Using the second equation we are telling a different story: As wealth increases test scores initially do as well but after a certain point increased wealth will hurt test scores. That picture isn't so rosy for the Bill Gates of the world!

And so we arrive at a key insight. By choosing our equations to tell a story, our inferences are in fact based on narrative modeling approaches. True, these inferences build upon numerous calculations and very advanced theoretical underpinnings, but ultimately what governs our conclusions and inferences are the stories or narratives we tell about our system. These are choices that we as narrators make and they not determined by an objective truth or reality.

Predictive Inference

Is there an alternative approach to inference that does not rely so heavily on narrative? Can we accomplish it without assuming the relationships between variables? The answer is yes. Although they are not often used, alternative prediction-based approaches to inference are available. In these approaches, rather than calculating statistical significances as a function of an assumed model, we calculate significances as a function of the simple question: "Does knowing X help us to predict Y?" This question is effectively identical to our earlier question – "Does X affect Y?" – but it is structured in an explicitly predictive manner. If the answer to the question is true, then we can say that there is a relationship between X and Y.

The techniques to accomplish prediction-based inference are much newer than classic techniques as linear regression as they rely upon extensive computing power and would not be possible without modern technology. One of these approaches is the *A3* method (XXX Citation) which uses resampling based algorithms to obtain estimates of predictive accuracy and statistical significance. *A3* focuses purely on predictive accuracy of a model to determine whether a variable is significant and often requires the automatic exploration of hundreds or thousands of competing models to find the one that best describes the data. The results of these analyses are inferences that are founded in the data of model fits only, not on subjective assumptions.

Predictive versus Narrative Modeling

As we can see, inferential techniques can be split into two categories: those based on narrative modeling methods and those based on predictive modeling methods. So – and this is a key advance – although there three categories of model purposes – prediction, inference, and narrative – there are only two fundamental approaches to constructing models: predictive modeling and narrative modeling.

This divide is not traditionally used in the modeling field, but it is truly at the heart of modeling. Understanding the distinction between these two types of modeling proves below to be much more valuable than mastering fine technical details. The choice of whether to build a predictive or a narrative model is a fundamental one that shapes every aspect of a model and determines its ultimate utility for a specific purpose. The following sections will describe these two types of models in more detail.

Predictive Models

How do we define a predictive model? The naive answer is that a predictive model is one that makes predictions. If a model generates predictions for a future outcome or a given scenario, than it must be a predictive model. By

this definition, a weather forecast is a predictive model as were the Obama administration's unemployment predictions we saw earlier.

Unfortunately, this straightforward definition is useless. Worse than being useless, it is actually quite dangerous.

Let us propose a model for next year's unemployment figures in the United States:

> Generate a random number from 0 to 1. If the number is less than 0.1, unemployment will be 20%. If the number is greater than or equal to 0.1, unemployment will be 0%.

There, we have just constructed a model of unemployment. Furthermore, our model creates predictions. With just a few calculations we can forecast unemployment for the coming year. Isn't that convenient?

Of course, this model is a joke. It is useless in predicting unemployment. However, using the naive definition of what it means to a predictive model, it would be classified as one.

What makes this simple model, such a poor model for prediction purposes?

There are several answers. We might start by saying it is too *simple*. If we are really trying to predict unemployment we should incorporate the current economic state and trends into our model. If the economy is improving, unemployment will probably drop and vice versa. This is a valid point. Let's address it by proposing an "improved" model:

> Generate a random number from 0 to 1. If the number is less than the percentage change in GDP over the past year, unemployment will be 20% plus the current unemployment rate. If the number is greater than or equal to 0.1, unemployment will be the net change in the consumer price index over the past 8 years.

Is this a better model? Clearly, it is more complex than the previous one and it incorporates some relevant economic data and indicators. Equally as clear, however, is that it is also a joke far from being a useful model.

These toy economic models show that just generating predictions is not a helpful criterion to define a predictive model. They also show that complexity and the use of relevant data is not a valid criterion. So how do we specify a predictive model? The answer is straightforward:

> A predictive model is a model that not only creates predictions but also must contain an *accurate assessment of prediction error*.

Read that statement again. The key point is that the assessment of prediction error must be accurate, which is different from the accuracy of the predictions themselves. Of course, ideally the predictions will be accurate; however this is often not possible. Many systems are governed to a significant extent by chance and no model, no matter how good it is, will be able to create accurate predictions for the systems.

If you know the level of prediction error, you can instead contextualize poorly fitting models. You can determine how much to discount their predictions in your decision-making and analysis. Furthermore, and this is crucial, you can compare different predictive models. If your current model is insufficiently accurate, you can develop another one and objectively test it to determine whether it is better than the current model.

Without measures of predictive accuracy, discussing predictions or comparing models that create predictions is an almost nonsensical endeavor. Such discussions will be governed by political concerns and partisanship as there is no objective foundation on which to base them.

Our two proposed models to estimate unemployment are thus clearly not predictive as no estimate of predictive error has been established. We can apply same this requirement to Obama's employment predictions we saw earlier. When we first presented the model, we called it a narrative model, which might have been slightly perplexing as the model did generate predictions. However, using our above definition of a predictive model we can see also that it is in fact not a predictive model. The model contains no estimate of prediction error (and one is not available in the original report) so it simply cannot be considered to be predictive.

If accurate estimates of prediction error are available, you can directly compare the prediction errors between different models to select the one with the lowest error. We could estimate prediction errors for the two joke models we proposed here along with the Obama administration's model to find the one with the lowest error. We would hope that the one the Obama administration presented to Congress would be the most accurate. Before we test it however, we must not make the error of fallaciously accepting a model to be good based on who presented it to us or its complexity.

Why do we so rarely hear about the predictive accuracy of models? There are numerous reasons but they boil down to three basic ones:

1. Assessing prediction error accurately is quite difficult.
2. Sharing prediction error may perversely decrease an audience's belief in a model.

3. Most models people use for prediction are in reality narrative models and their predictive error is either abysmal or irrelevant.

Let's look at each point in detail. First consider the issue of the difficulty of assessing prediction error. In general, obtaining an accurate assessment of prediction error is much more difficult than developing the predictions themselves. Most commonly used approaches (for instance the standard R^2 from linear regression) have significant flaws. There are both theoretical and numerical methods that can be used to make more accurate prediction errors in many cases (this will be discussed further in the section the Cost of Complexity; see also @FortmannRoe:2012tf). When dealing with time series data, however, like most of the models explored in this book, it is often almost impossible to accurately assess model prediction error. Recently, theoretical technique to approach these issues have just begun to be developed (e.g. @He:2009jp or @King:2008jq) but they are still impractical to apply in many cases so far.

If the challenge of measuring prediction error is surmounted, there is an even more formidable barrier to its being published with the model. There is a perverse phenomena that the act of reporting prediction error can *decrease* the confidence an audience gives a model. An anecdote was relayed to us by a member of a team working on a model of disease spread. His team shared the predictions from the model with a group of policy-makers. Everything was going fine until the audience saw the error bars around the predictions. Where his audience had been content with the raw predictions, they were quite unhappy with the predictions when accompanied by their accurately estimated uncertainties. Why was this? Was the team's model particularly bad or did these policy-makers have a better model at their disposal? No. In a world where policy-makers and clients are constantly shown models (like Obama's unemployment figures) with no measure of uncertainty (or even worse, poorly calculated, artificially low uncertainty), they come to have unrealistic expectations and often turn away good science in favor of magical thinking.

Finally, the most likely reason supposedly predictive models do not include prediction error is that they simply are not predictive. We have seen how models developed for a purportedly predictive purpose can actually be narrative models in disguise. Just why is this too often the case? You need only look at the reason for most modeling projects. It is very rare that models are commissioned solely for the purpose of generating an accurate prediction. Frequently, models are part of some political process within an organization or across them (whether an organization be a for-profit company or a non-profit such as a university). Ultimately, the people funding the model expect it to prove a point to their benefit. In environments like these, it is to be expected that some predictive modeling efforts will be sidetracked by political concerns or compromised in the process.

We can see the results of such influences in the predictions generated for unemployment presented earlier. Figure 3 shows the projections for the unemployment

rates with and without the stimulus plan just as in Figure 2. Overlaid on this are now the true values of unemployment the occurred after the predictions were made. As is readily evident, the original modeling and predictions were well off the mark. Not only was reality worse than the projections assuming the stimulus was enacted (which it was) it is much worse than the projections for the economy assuming the stimulus had never been enacted at all! This is just a small example – one that is sadly replicated over and over again in business and policy-making – of mistakenly treating a narrative model as a predictive one.

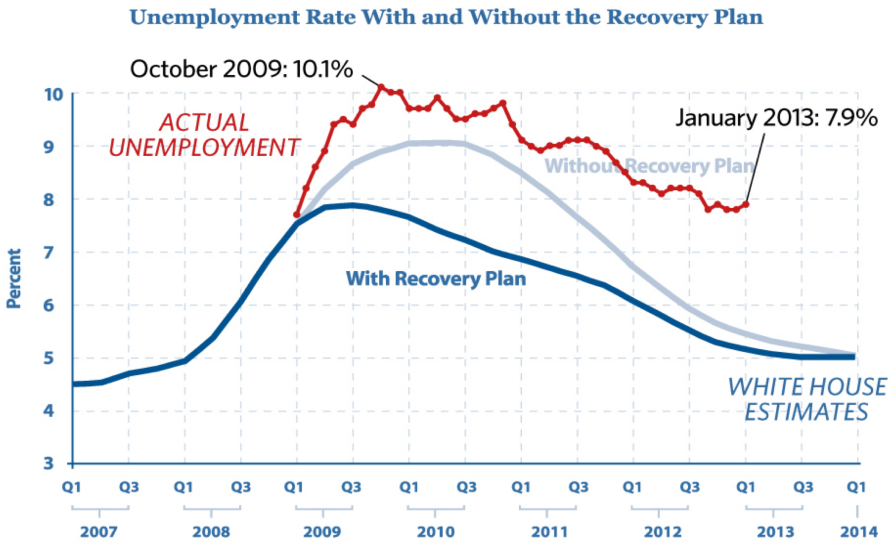


Figure 3. Unemployment predictions versus reality [@TheHeritageFoundation:2013vu]

Narrative Models

In contrast to predictive models, a narrative model is one built to persuade by telling a story. When many people first hear the "narrative" terminology, they respond negatively. "It's just a story." We find this strange, as narratives are the fundamental human form of communication. We tell narratives to our friends and relatives. Politicians communicate their policies to us using narratives. Of course the vast majority of our entertainment is focused on narratives[4]. Business leaders and managers attempt to describe their strategies to us using story lines; and business books are in general dominated by anecdotes plotted along the way to making their points.

[4]Even sports, a form of entertainment that innately contains no narrative, becomes wrapped in narrative as the announcers and commentators attempt to create stories to engage us.

We as a species do not view the world as a collection of numbers and probabilities; instead we see consequence and meaning. In short, narratives are how we see the world.

One critique of the term narrative is that it lacks numbers, quantified data, or mathematics. This could not be further off the mark. There are many ways to construct narratives. Words are one, pictures are another, and music is a third. Numbers and mathematics are just another way of telling a story.

In fact, most statistical and mathematical models are infused with narrative models. We looked earlier at the case of linear regression as a tool to predict test scores as a function of wealth. Again the mathematical equation for this simple model was:

$$Score = \beta_0 + \beta_1 \times Wealth$$

This equation defines a narrative. Translating this narrative into words, we would say:

> Test scores are only determined by the wealth of a student's family. A child whose family is broke will have a test score, on average, of β_0. For every dollar of wealth a child's family accumulates, the child will score, on average, better on tests by β_1.

You might or might not agree with this storyline (in our view it is a nonsensical and reductionist view of child achievement) but it shows the strict equivalence between this mathematical narrative and narrative prose. This process can be applied to all mathematical models. The mathematical definition of the model can be converted directly, with more or less lucidity, into a story describing how the system operates. The same can also be done in the reverse: we can take a descriptive narrative of a system and convert it into a mathematical description. As we have seen (will see? XXX) this is what tools like reference modes and pattern matching are designed to do efficiently: elicit a narrative from a subject in a way which can be reformulated quantitatively.

The question of how to assess the quality of a narrative model is an important one. With predictive models, we can compare competing models based primarily on predictive accuracy[5]. But how do we evaluate and compare the quality of narrative models?

The key criterion in assessing a narrative model is its ability to be *persuasive*. Although persuasion is not an objective measure in the same sense prediction accuracy is, we can decompose persuasiveness into two components for our

[5]Other criteria include ease of use, cost of filling data requirements, and computational requirements. But all those are generally secondary to prediction accuracy.

purposes: believability and clarity. A persuasive model is one that is both believable and effectively communicates its message.

When building a narrative it is very important to use tools that are well suited to meeting these components. Unfortunately, many statistical models, including regressions, are poorly suited to this two-fold task in many ways. Most statistical models depend on unrealistic and highly technical assumptions about the data. If these assumptions were enumerated in plain English, they would often conflict with people's understanding and in fact end up discrediting the model. The "alternative" has been to leave these assumptions hidden creating a black box model opaque to outside inspection.

This is a shame in our view. Such a stratagem can be successful if the authority presenting the model is prestigious enough. But the misdirection will quickly fail if any kind of rigorous scrutiny is applied to the model. Narrative models should never be given any real credence if the operation of the model is not transparent. Most statistical models are built on assumptions that are never made transparent to the audience.

The modeling techniques presented in this book, on the other hand, are well suited for narrative modeling. The techniques we present are "clear box" modeling where the workings of the model are transparently evident and accessible. Our models have their structure explicitly described using an accessible modeling diagram showing the interactions between the different components in the model. The equations governing the model's evolution are clear and readily available for each part of the model[6]. Furthermore, these modeling techniques used here make it straightforward to generate animated illustrations and displays to clearly communicate model results.

Synthesis

Now that we have thoroughly described the concepts of narrative and predictive models we can conclude this chapter by taking a step back and reemphasizing that these two categories do to represent specific modeling techniques. You can build a stock and flow model to tell a story about a system resulting in a narrative model. If your story of the system accurately represents how the system operates in reality, then you will also have a model that generates accurate predictions.

Similarly, you can apply a linear regression to a dataset. If the relationship in the data is truly a completely linear one, then the result of this regression will be the most accurate predictive model you could build. On the other hand, if you do not assess the predictive accuracy of the model and just use a linear regression because it is easy to interpret or because it matches you understanding of reality, then you have a narrative model.

[6] Admittedly, for complex models it may still require a significant investment on the part of an audience to fully understand the logic and equations in the model. But the opportunity is available.

The key criteria to remember when building your own models or assessing other peoples models is that a predictive model is one for which you have an accurate assessment of the errors of the predictions. A good predictive model is one that has low relative errors when compared to other predictive models for the same system. A narrative model is one that tells a story about the system. A good narrative model is one that persuades an audience that its story is credible.

Chapter 8

Building Confidence in Models

When used correctly, the transparency of the modeling techniques presented in this book results in models that are powerful persuasive tools. As with any model, however, there are concerns and questions will invariably be raised which could cause users to doubt the result of the modeling work. There are a number of techniques that you can use to help preemptively address these concerns and increase an audience's confidence in your model.

The idea of building confidence in a model is closely tied to the standard concept of model verification and validation. We dislike this conceptual approach to assessing models as it seems to imply that a model can go through a process to get a big fat "VALID" or "VERIFIED" stamp on it. Returning to Box's quote that "all models are wrong, but some are useful", in reality all models are wrong and none of them are completely valid, period. Models can however be useful, especially narrative models in which the audience has confidence.

We favor the conceptual approach put forth by @Forrester:1978vy, that there is not any single test or suite of tests that will verify or validate a model and that validity should instead be thought of as a function of confidence. This is a view that differs from that held by some modelers and laypeople. As Forrester and Senge note, "the notion of validity as equivalent to confidence conflicts with the view many seem to hold which equates validity with absolute truth." We share their belief that model confidence is built up piece by piece from a variety of tests that, though they cannot prove anything, together comprise a persuasive case for the quality of a model.

There are three distinct areas where confidence needs to be developed:

1. That the model itself is well designed.
2. Given a design of the model, this design is implemented correctly.
3. The conclusions drawn from the model are accurate.

In the remaining sections of this chapter we will look at each of these different areas in detail. We will explore the different tests and tools that can be used to build confidence for each area.

Model Design

Fundamentally the design of a narrative model is of utmost importance and needs to be justified to an audience[1]. There are two primary aspects to a model's design design: the structure of the model and the data used to parameterize the model.

Structure

The structure of the model should mirror the structure of the system being simulated. Depending on the system complexity, the model structure may need to carry out more or less aggregation and simplification of this reality. Nevertheless, all the primitives in the model should map on to reality in a way that is understandable and relatable to the audience. Thus if there is some object in the real system that behaves as a stock, a stock should exist in the model mirroring the object's position within the system. The same should hold true with the other primitives in the model. Each primitive would ideally be directly mappable onto a counterpart in the real system and any key component in the real system should be mappable onto primitives in the model. Furthermore, feedback loops that exist in the system should exist within the model. These feedback loops should be explicitly identifiable in the model and would ideally be called out or marked in a way that highlighted their presence to an audience.

Furthermore, the model structure should include components that an audience thinks are important drivers of the system. Missing a factor that the audience considers to be a key driver can fatally discredit a model in an audience's mind irrespective of the performance or other qualities of a the model. This is true even if the factor has in fact a negligible effect. Generally speaking, it is much easier to include a factor an audience views as important than it is to later on convince the audience that the factor does not in actuality matter.

Data

The more a model uses real-world data, the more confidence an audience will have in the model. Ideally, you have empirical data to justify the value of every primitive in your model. In practice, such a goal may be a pipe dream. Indeed, for a complex model, obtaining data to parameterize every aspect of it is usually

[1] This is different from predictive models where the results of the model are much more important than the design and the "proof is in the pudding"" so to speak.

impossible[2]. When faced with model primitives that do not have empirical data to parameterize them, an approach must be taken to ensure that it does not appear that their values were chosen without justification or to arrive at a predetermined modeling conclusion. Sensitivity testing, as discussed later on, is one way to achieve this. Another is to carry out a survey of experts in the field in order to solicit a set of recommended parameter values that can then be aggregated or used to justify the ultimate parameterization.

Peer-Review

Going through a peer-review process can be extremely useful in establishing confidence in a model. Two general types of peer-review are available. In one, the model may be incorporated into an academic journal article and submitted for publication. The article will then peer-reviewed by generally two or three anonymous academics in the field who critique it and judge whether or not it is a worthy contribution to the literature, thus meriting publication. In the second type of peer-review, a peer-review committee may be assembled (hired) to review a specific model and provide conclusions and recommendations to clients.

Peer-review can be very useful in establishing the credibility of a model. A credible model is a model one can be more confident in, other things being equal. By engaging an independent group of exports to assess the model, their conclusions about its quality have the appearance of greater validity than those of the self-interested modelers[3]. This can be especially useful when trying to meet some abstract standard such as that the model represents the "best available technology" or the "best available science".

A key risk of a peer-review is, of course, that the peer-review members will find a model deficient in important respects. Good criticism can be very useful and help improve a model. However, some criticism received in practice may be nitpicking details or detrimental advice that would make the model worse if followed.

Model Implementation

Although it is not as much a lightning rod as is model design, the implementation of a model specification is a point where significant error can occur. Programming mistakes or mistyped equations can introduce bugs into a model that can be hard to identify later on. This is a particular problem in black-box models but it is still an important point to consider for all types of models

[2]Leading to the clichéd conclusion of many modeling studies: "We are unable to draw strong conclusions from this modeling work. Instead, our contribution has been to show where additional data needs to be collected."

[3]When the peer review panel is hired by the client there is some conflict of interests, but the panel members should not be swayed by this.

including those presented in this book. Fortunately, a number of steps can be taken to ensure the model is implemented correctly.

Primitive Constraints

For many of the primitives in the model, there will be natural constraints. For instance, a stock representing the volume of water in a lake can never fall below 0. Similarly, if a variable represents the probability of an event occurring, it must be between 0 and 1.

Often these constraints are implicit without being formally specified in the model. A modeler may think, water volume can never become negative so why would I need to specify it? However, the existence of these constraints provides an opportunity to implement a level of automatic model checking. By specifying that a primitive can never go above or below a value (using the **Max Value** and **Min Value** properties in Insight Maker), you can create in effect a canary in the coal mine that warns if something is wrong in the model. If these constraints are violated an error message can be given letting you know that you need to correct some aspect of your model.

This concept of constraints in models is similar to the concept of "contracts" which are support in some programming languages. These contracts define and constrain the interaction between different parts of the program causing an error to be generated if the contract is violated. The Eiffel programming language probably has the best support for this approach to development.

Unit Specification

Since we introduced units in Chapter 3, we showed that they could be a useful tool in constructing models. Units can also be used to ensure that equations are entered correctly. If you fully specify the units in a model, many types of equation errors will result in invalid units, which will create an immediate error. By employing units in your model you can automatically catch a whole class of errors and mistyped equations.

Regression Tests

Other tests than those specified above can be developed. For instance, the proper behavior of one part of the model may be determined and automated tests created to periodically confirm that the model continues to exhibit the correct behavior. Development of such tests are a common part of software engineering that we wish would see more use in model development. Insight Maker itself has a suite of over 1,000 individual regression tests that automatically test every aspect of its simulation engine.

In regards to regression testing, it is important to ensure these tests are automated. It is not enough to examine a portion of the model, determine it is currently working correctly, and leave it at that. The problem is that future

changes may break the existing functionality (i.e. a "regression", the introduction of an error or reduced quality compared to an earlier version of the model). Especially for complex models, a change in one part of the model may have an unexpected effect in another part. By implementing a set of automatic checks, you can protect your model against unintended changes and regressions.

George Oster and his class XXX

A Second Pair of Eyes

That is not to say, however, that spot and point-in-time checks are not worthwhile. It can be very useful to have a second modeler review your models and cross-check the equations. This helps not only to check simple mistakes but also to question and critique the fundamental structure and choices of the model.

The gold standard in verifying that a model is implemented correctly according to specification is to have a second modeler completely reimplement the model according to that specification. Such reimplementation should ideally occur without access to the original model's code base to ensure that the second modeler does not simply copy bugs from the original model into the reimplementation. If the results from the two implementations concur, that is strong evidence that the model has been implemented correctly. Although potentially an expensive exercise, it will also most likely identify numerous ambiguities in the specification, which could be valuable in and of itself.

Model Results

Given that the design of the model and its implementation are assumed to be correct, the burden still falls upon the modeler to transfer her confidence in the model's results to her audience. There are several different ways this can be done.

Expected Results

The first way is to demonstrate that the model generates expected results for normal inputs. For instance, if you had a model a reservoir, you would expect the volume of the reservoir to decline over time during the summer due to evaporation if no more water flowed into it. You can additionally test extreme scenarios and show that they generate the expected results. If, for example, your reservoir were empty, you would expect the amount of water evaporating from it to be zero. By enumerating these standard cases and showing the model results match the expected results you can help build confidence in the model.

Often these expected results can be described in terms of a curve showing how the values of one of the stocks or variables in the system is expected to change over time. This curve can be taken from historical data (a reference behavior pattern), or simply drawn on a piece of paper by experts familiar with the system (an excepted behavior pattern).

Counterintuitive Results

Another attempt to increase confidence in a model is to show unexpected results that are justifiable. Imagine a model that for a certain set of inputs would create what, at first glance, appeared to be the "wrong" behavior. Some lever in the model could lead to unexpected results. When first shown these results, they could decrease an audience's confidence in the model. If the audience was then walked through the model step by step to show how those results proved to be correct and mirrored reality, then that could well increase their confidence in the model results.

Forecasting

Possibly the most persuasive action to convince an audience of the effectiveness of a model is to forecast the future and then to show this forecast to be correct. This, of course, is difficult to do in practice for multiple reasons including the fact that the scale of a model is often such that it could take several years or decades to generate data to test the model. Additionally, it must be remembered that most narrative models are poor predictors and should not be used for predictive purposes solely.

Sensitivity Testing

Sensitivity testing is a broad field that has the potential to address many questions and doubts that may arise about a model. In general, the variables and numeric configuration values in a model will never be known with complete certainty. When the results from an election poll are published, the pollsters publish not only their predictions but also the uncertainty in the prediction (e.g., "the Democratic candidate will obtain $52\% \pm 3\%$ of the vote"). Similarly when a building is constructed, the materials used will have certain properties – such as strength – that again are only known up to some errors or tolerance. It is the engineer's and contractor's responsibilities to ensure that the materials are sufficient even given the uncertainty of their exact strengths.

The same occurs when modeling. Most primitive values in the model will have to be estimated by the modeler and there will be an error associated with these values. Of course the error will also be propagated through the model when it is simulated and affect the results generated by the model. This error is one factor that can create doubt about a model and reduce an audience's confidence.

As a modeler, one approach to address this doubt would be to try to measure all the model's variables with great accuracy. You could search the available literature, undertake a meta-analysis of current results, carry out new experiments, and survey experts to get as precise a set of parameter values as possible. If you were able to say with strong certainty that these values were so accurate and the errors so small that their effect on the results in negligible, then that would be one way of addressing the issue of uncertainty.

However, all of this is often impossible to do. When dealing with complex systems it is almost always the case that at least a couple variable values will never be known fully with certainty. In this case, no matter how much research or experiments you do, you will never be able to pin down the precise values of these variables. How do we handle these cases?

The answer is straightforward: Rather than trying to eliminate the uncertainty, we embrace it by explicitly including it in the model. If you can then show that the results of your model do not significantly change even given the uncertainty, you have a persuasive case for the validity of your results. Of course the results will always change when the uncertainty is introduced, but if the model conclusions persist even in the face of this uncertainty it will greatly increase your audience's confidence in the results.

Uncertainty can be explicitly integrated into a model by replacing constant primitive values with a construct that represents the uncertainty in that value. Imagine you had a simple population model of rabbits in a cage. You want to know how many rabbits you will have after two years. However, you don't know how many rabbits there initially are in the cage. You have been told that there are probably 12 rabbits, but the true number could range anywhere from 6 to 18.

If you model your population as a single stock, what should the initial value be? A naive model could be built where you the initial value of the rabbit stock was specified as 12. However, that does not incorporate the uncertainty and could be a source of criticism or doubt for the model. An alternative would be to specify that the initial value of the stock is a random number with a minimum value of 6 and a maximum value of 18. So each time you run the model you will get a different result. If you ran the model once, the initial value might be chosen to be 7 and you would obtain one result. If you ran the model again, the initial value might be 13 and you would get another result.

If you run this stochastic model many times, you obtain a range of results. These results can be automatically aggregated to show the range of outputs. For instance if you ran the model 100 times you could see what the maximum and minimum final populations were. This would give you a good feeling for how many rabbits you needed to prepare for after two years. In addition to the maximum and minimum you might be interested in the average of these 100 runs: the expected number of rabbits you would see. You could also plot the distribution of the final population sizes using a histogram to see how the results are distributed. This distribution would show how sensitive the outputs are to the uncertainty in the inputs: a form of sensitivity testing.

There are four key distributions that are useful for specifying the uncertainty in a variable:

Uniform Distribution : The uniform distribution is defined by two parameters: a minimum and a maximum. Each number within these two boundaries has an equal probability of being sampled. The uniform distribution is useful when

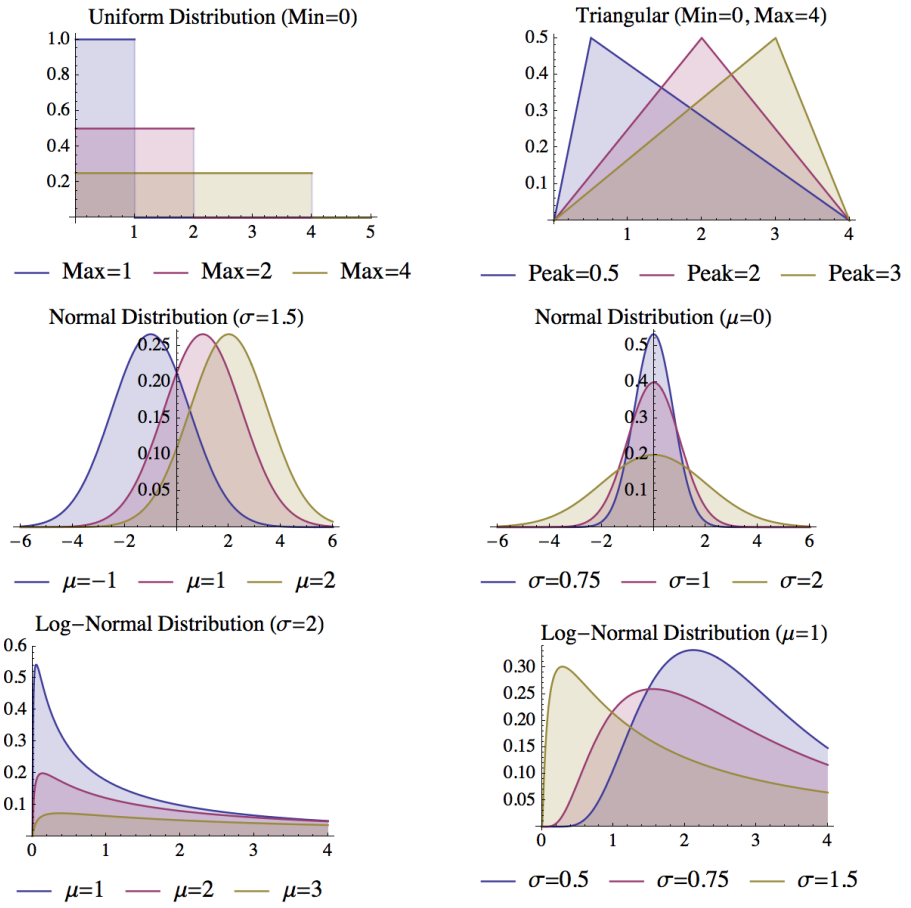


Figure 1. Common Distributions for Sensitivity Testing with Sample Parameter Values

you know the boundaries on the values a variable can take on, but you do not have any information on the likelihood of the different values within this region. The uniform distribution can be used in Insight Maker using the function `Rand(Minimum, Maximum)`, the two parameters are optional and will default to 0 and 1 if `Rand()` is called without them.

Triangular Distribution : The triangular distribution is defined by three parameters: the minimum, the maximum, and the peak. Like the uniform distribution, the triangular distribution will only generate numbers between the minimum and maximum. Unlike the uniform distribution, the triangular distribution will not sample all numbers between these boundaries with equal likelihood. The value specified by the peak will have the most likelihood of being sampled with the likelihood falling off as you move away from the peak towards either

the minimum or maximum boundary. The triangular distribution is useful when you know the both the most likely value for a variable and you also know boundaries for the values a variable can take on. The triangular distribution can be used in Insight Maker using the function `RandTriangular(Minimum, Maximum, Peak)`.

Normal Distribution : The normal distribution is defined by two parameters: the mean of the distribution (generally denoted μ) and the standard deviation of the distribution (generally denoted σ). The most likely value to be sampled from the normal distribution is the mean. As you move away from the mean (in either a positive or negative direction), the likelihood of a number being sampled decreases. The standard deviation controls how fast this likelihood falls as you move away from the mean. Small standard deviations result in steep declines in the likelihood while large standard deviations result in more gradual declines. The normal distribution is useful when you do not have boundaries on the values for a variable but you do know what the most likely value for the variable should be (the mean). The normal distribution can be used in Insight Maker using the function `RandNormal(Mean, Standard Deviation)`.

Log-normal Distribution : The log-normal distribution is closely related to the normal distribution. In fact the logarithm of the values samples from a normal distribution will be log-normally distributed. Like the normal distribution, the log-normal distribution is defined by two parameters: the mean and standard deviation. Where the log-normal distribution differs from the normal distribution, is that negative values will never be generated by the log-normal distribution. Thus it is useful when you have a variable which you know cannot be negative but for which you do not have an upper bound. The log-normal distribution can be used in Insight Maker using the function `RandLogNormal(Mean, Standard Deviation)`. The log-normal distribution can also be used to represent other types of one-sided boundaries. For instance, the following equation could be used to represent a variable whose number was always less than 5: `5-RandLogNormal(2, 1)`

There are many other forms of probability distributions. Some notable ones are the Binomial Distribution (`RandBinomial(Count, Probability)`), the Negative Binomial Distribution (`RandNegativeBinomial(Successes, Probability)`), the Poisson Distribution (`RandPoisson(Lambda)`), the Exponential Distribution (`RandExp(Lambda)`) and the Gamma Distribution (`RandGamma(Alpha, Beta)`). These distributions can be used to address very specific modeling usage cases and needs (for instance, the Poisson distribution can be used to model the number of arrivals over time), however, the four distributions described in detail above should generally be sufficient for most sensitivity testing needs.

When important practical tip when using sensitivity testing within the System Dynamics context is to be careful about specifying random numbers within variables. The value of a variable is recalculated each time step. This means that if you have a random number function in the variable, a new random value

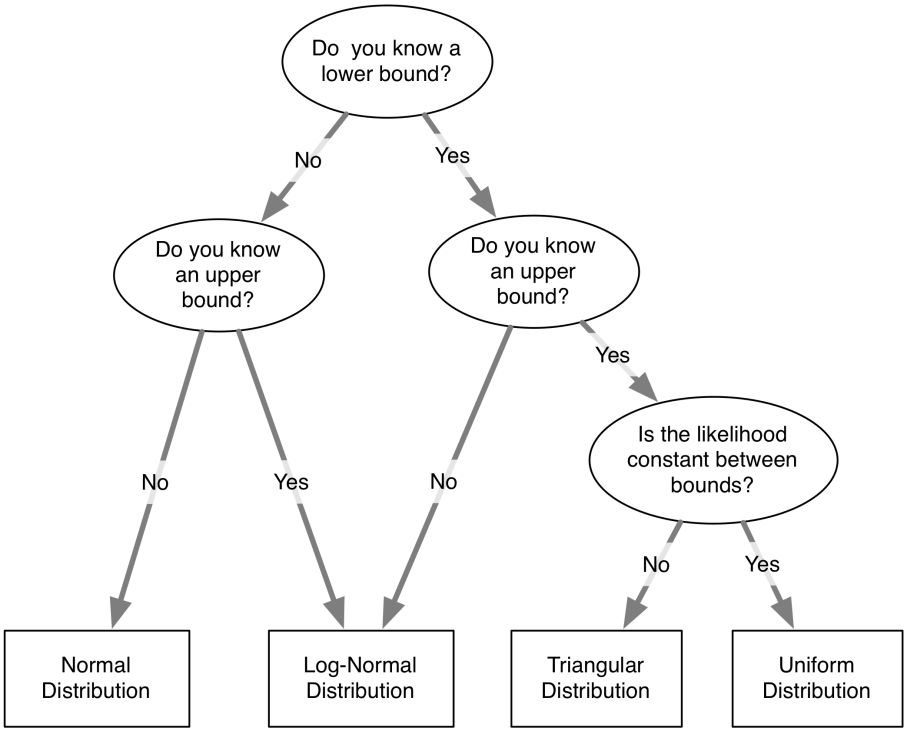


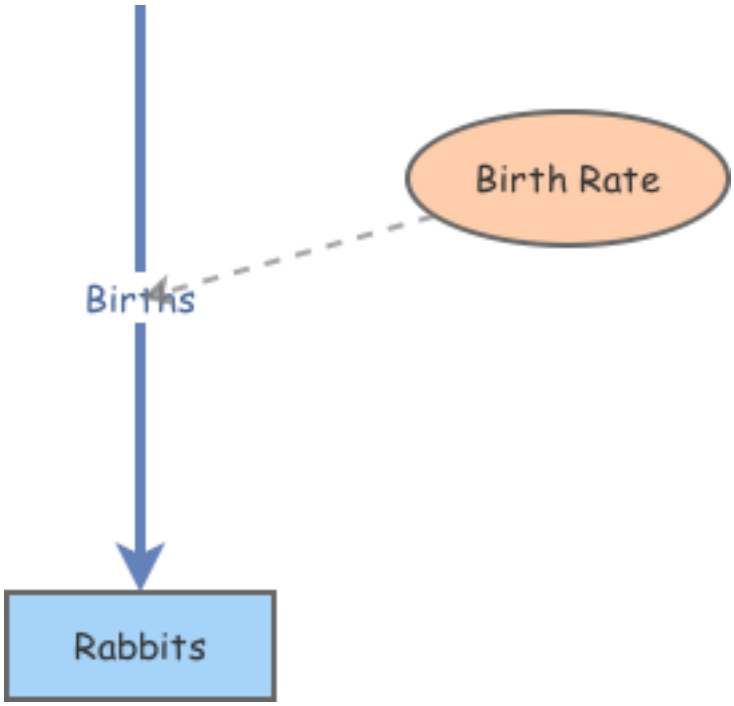
Figure 2. Choices in Selecting a Distribution for a Variable's Value

will be chosen each time step. This can create a problem if the random value is supposed to be fixed across the course of the simulation. For instance, we may not know the birth rate coefficient for our rabbit population, but, whatever it is, we assume it is fixed over the simulation.

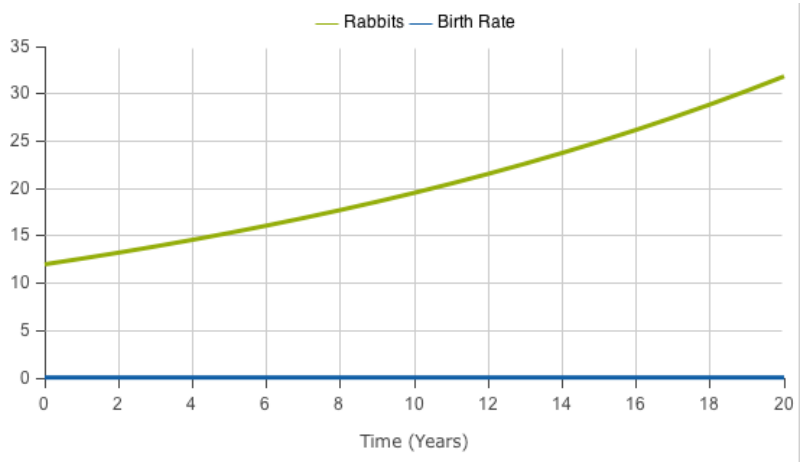
A simple way to handle these fixed variable values would be to replace the variables with stocks. The stocks initial value could be set to the random value and it would only be evaluated once at the beginning of the simulation and kept fixed thereafter. This approach, though very workable, however violates the fundamental metaphors at the hard of System Dynamics. In Insight Maker, another approach is to use the `Fix()` function. When used with one argument, this function evaluates whatever argument is passed to it a single time and then returns the results of that initial calculation for subsequent time steps. So instead of having the simple equation `Rand(0, 10)` in a variable to generate a random number between 0 and 10, you could place `Fix(Rand(0, 10))` in the variable. The first equation would generate a new random number each time step, the second equation will generate one random number and keep it constant throughout the simulation.

Sensitivity Testing

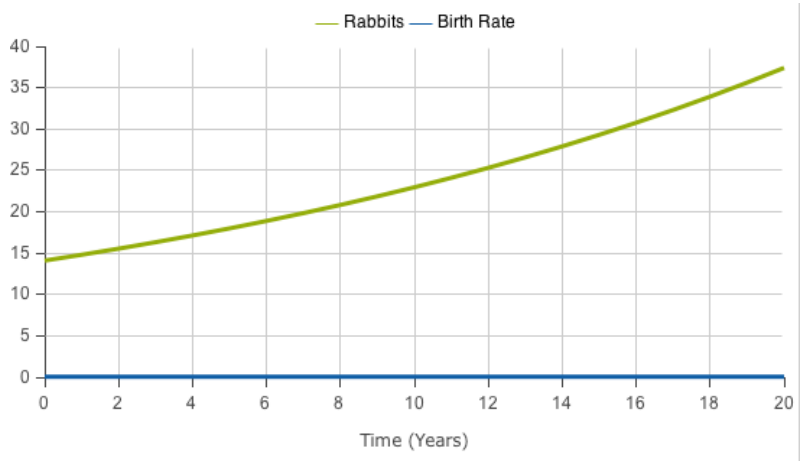
1. Let's illustrate the usage of sensitivity testing using our rabbit example. First we will construct a simple exponential growth model.
2. Create a new **Stock** named [**Rabbits**].
3. Change the **Initial Value** property of the primitive [**Rabbits**] to `12`.
4. Create a new **Flow** going from empty space to the primitive [**Rabbits**]. Name that flow [**Births**].
5. Create a new **Variable** named [**Birth Rate**].
6. Change the **Equation** property of the primitive [**Birth Rate**] to `0.05`.
7. Create a new **Link** going from the primitive [**Birth Rate**] to the primitive [**Births**].
8. Change the **Flow Rate** property of the primitive [**Births**] to `[Birth Rate]*[Rabbits]`.
9. The model diagram should now look something like this:



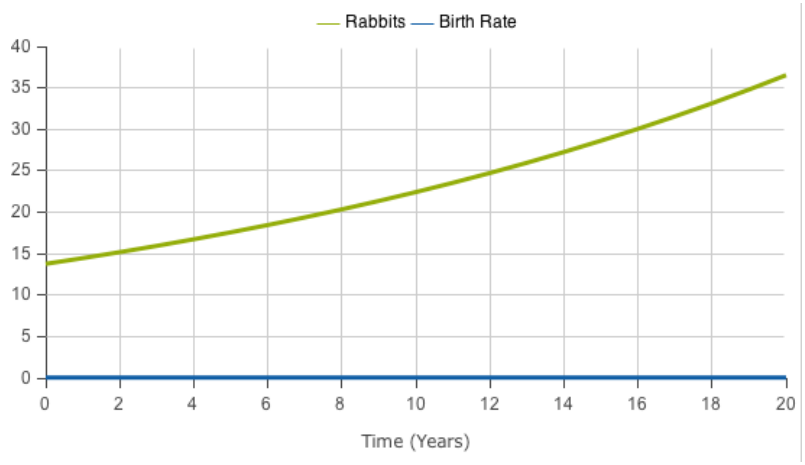
10. This is the basic outline for the model. We assume a fixed value of 12 rabbits and a fixed birth rate of 0.05.
11. Run the model. Here are sample results:



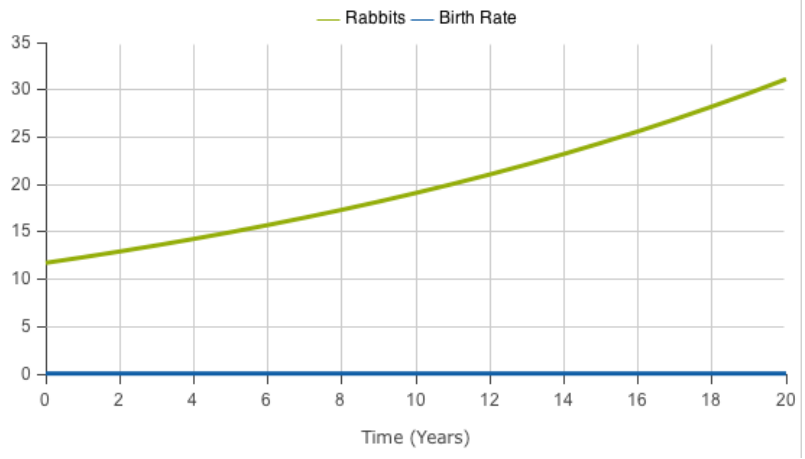
12. When we simulate we obtain the same results each time.
13. Change the **Initial Value** property of the primitive [**Rabbits**] to `RandTriangular(6, 18, 12)`.
14. Now, let's try to incorporate uncertainty. Given that we know that there can be between 6 and 18 rabbits initially with an expected value of 12, we can use the **RandTriangular()** function to model this.
15. Change the **value** property of the primitive [**Birth Rate**] to `RandLogNormal(0.05, 0.03)`.
16. We also do not know the birth rate with certainty. We know, however, that the rate must be greater than 0, and lets say we can assume the expected value is 0.05. We can use the **RandLogNormal()** function to model this type of uncertainty.
17. Run the model. Here are sample results:



18. Run the model. Here are sample results:

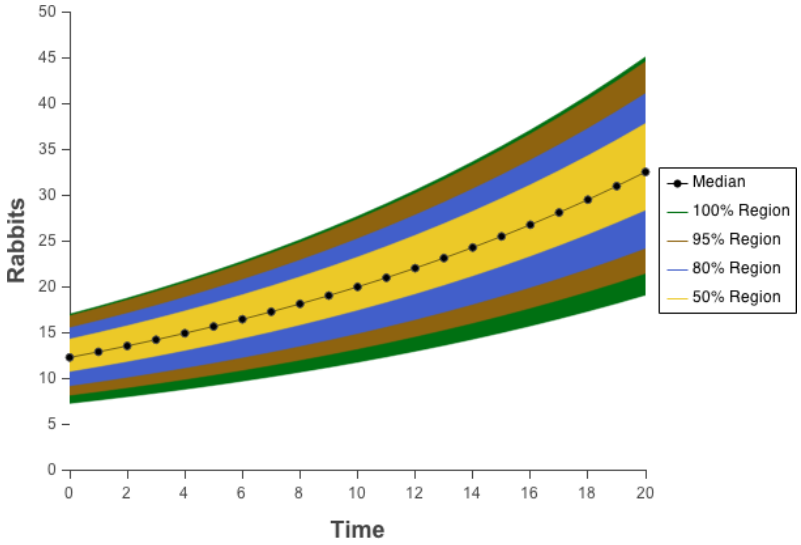


19. Run the model. Here are sample results:



20. Now, we can simulate this mode a few times and see that each time we run the simulation we get a different result.

21. We can now use sensitivity testing to see the range of results given this specified uncertainty. We'll do 100 runs of the model and aggregate the results to see the expected distribution

22. 

23. We can readily see the range of results allowing us to make decisions incorporating our known uncertainty about parameter values.

The astute reader will notice that our discussion up to this has failed to address an important point: how do we determine the uncertainty of a variable? It is very easy to say that we do not know the precise value of a variable, but it is much more difficult to define the uncertainty of it. One case where we can precisely define uncertainty is when you take a random sample of measurements. For instance, suppose our model included the height of the average American man as a variable. We could randomly select a hundred men and measure their heights. In this case our uncertainty would be normally distributed with a mean equal to the mean of our sample of one hundred men and a standard deviation equal to the standard error of our sample of one hundred men[4]. For any random sample of n values from a population, the same should hold true: you will be able to model your uncertainty using a normal distribution with:

$$\mu = \frac{Value_1 + Value_2 + Value_3 + ... + Value_n}{n}$$

[4]Please note that this contradicts slightly what we said earlier. Clearly, a person cannot have a negative height while the normal distribution will sometimes generate negative values. So wouldn't a log-normal distribution be better than a normal distribution? Mechanistically, it would, however statistically we can show that due to the Central Limit Theorem the normal distribution does asymptotically precisely model our uncertainty. Given a large enough sample size (100 is more than enough in this case), the standard deviations for uncertainty will be so small that the chances of seeing a negative number (or even one far from the mean) are effectively none.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Value_i - \mu)^2}$$

However, in most applied cases you will not be able to apply this normality assumption. Generally you will not have a nice random sample, or you might have no data at all and instead have some abstract variable you need to specify a value for. In these cases, it is up to you to make a judgment call on the uncertainty. Choose one of the four distributions detailed above and use whatever expert knowledge available to you to place an estimate on the parameterization of uncertainty. One rule of thumb, however, is that it is better to overestimate uncertainty than underestimate it. It is better to err on the side of overestimating your lack of knowledge than it is to obtain undue confidence in model results due to an underestimation of uncertainty.

Confidence and Philosophy

The quality of a model in the eye of an evaluator is to a significant extent influenced by the worldview (more or less coherent and the philosophical orientation (if any) of the evaluator. Broad world-views or epistemologies[5] exist. One key divide in different epistemological theories continues to be between those theories that contain a strong belief in a concrete true reality that our knowledge can accurately capture and those theories that believe our knowledge is partially or wholly independent from reality.

Epistemological theories that are primarily in the first camp are those such as positivism or empiricism. Theories in the latter camp include constructivism and idealism. Constructivism is a popular theory of knowledge that claims knowledge is constructed with social context and historical time. Our presentation of confidence building for narrative models in this chapter is implicitly in line with a constructivist theory of knowledge.

In our discussion of confidence building we repeatedly refer to matching the beliefs of the audience. We recommend creating simulations and behavior in our models that match an audience's expectations for the behavior of the system. This is distinct from saying that you should match the reality of true system. Ideally, true behavior of the system and an audience's mental models of the system should be equivalent, but in practice they may well differ. Although confidence in a model will be boosted by strictly matching the mental models of an audience, a truly effective narrative model should be persuasive enough to change the mental models of an audience.

Our discussion of predictive models from the previous chapter does not fall within a constructivist world-view as we are claiming that there are objective

[5]From the Greek word "epistēmē" meaning "knowledge" or "understanding", epistemology is the branch of philosophy describing how we understand or come to know the world around us.

"outside-ourselves" measures of predictive accuracy we can obtain. It should go without saying that predictive models may not be accurate reflections of reality, even in their own terms. The mathematics of a predictive model may be unrelated to the true system that is being modeled, yet is may still create accurate predictions. As such, our discussion of predictive models is not really a positivist or empiricist one. Instead this discussion would fall under the epistemological theories of pragmatism or instrumentalism which claim a theory or model should be assessed on how well it predicts which may be independent of the truth of the theory itself.

Chapter 9

The Process of Modeling

Chapter to go here...

Pattern Oriented / Event Oriented

From models to insights

KIDS/KISS

Sterman 93/94 "learning in a complex of world"

Confirmation bias sterman, can't trust people to build models

Chapter 10

The Mathematics of Modeling

This chapter takes the modeling techniques introduced earlier in this book and places them within a firm mathematical framework. The contents of this chapter are quite technical in parts and to fully understand them requires knowledge of basic calculus and linear algebra. We present the material because it is important for both readers who want a deep understanding how their models operate and also those who wish to understand how System Dynamics fits within the larger field of mathematical modeling. For users who approach systems thinking and modeling from a more qualitative angle, this material may be browsed or safely skipped.

Differential Equations and System Dynamics

Differential equations are a common mathematical tool used to study rates of change. Some basic terminology needs to be learned in order to discuss differential equations. After introducing this new terminology, we will then tie it back to the modeling techniques you've already learned.

State Variable : A state variable is an object that represents part of the state of a system. For instance, in a population model you could have a state variable representing the current number of individuals in that population. In a model of a lake, you could have a state variable representing the current volume of water in the lake. In equations, state variables are often represented using Roman letters such as X, Y or Z.

Derivative : Derivatives define rates of change in state variables. For instance, if we had a state variable representing the size of a population, a derivative would specify how this population grows or shrinks over time. The population's derivative would aggregate all changes such as births, deaths and immigration or emigration to show the net change in the state variable over time. Similarly, in the case of a model of a lake, the lake volume state variable would have a derivative showing how much net water flows into or out of the lake over time.

Given a state variable X, the derivative of X with respect to time is generally written as dX/dt but can also be written as X' or $\dot{X}$.

Let's put this new terminology to work to define a simple model. We start by creating an exponential growth population model. We only need one state variable in this model to represent the size of the population. We denote this state variable as P. We need to define one parameter to control the growth rate in the population. We will denote this growth rate parameter α.

The resulting differential equation exponential growth model can be written simply as:

$$\frac{dP}{dt} = \alpha \times P$$

This indicates that the rate of change for the population for one unit of time is $\alpha \times P$. Our model is not quite fully specified yet as we do not know what the initial value of the population is. Differential equation models are often additionally specified by providing the values of the state variables at a specific point in time. Below we indicate that the population size at time 0 is 100.

$$P(0) = 100$$
$$\frac{dP}{dt} = \alpha \times P$$

You may have already noted that this model is easy to construct using the techniques we have already introduced in the book. In fact we have discussed this type of model several times. We could construct it with System Dynamics tools using a stock to represent the population (P), a flow to represent the change of population (dP/dt) and a variable to represent birth rate (α). We could specify our initial condition of a population size of 100, by setting the initial value for the stock for 100.

This is an important point. Many differential equation models[1] can be directly represented using the System Dynamics modeling techniques described in this book. Similarly, a System Dynamics model can be rewritten as a differential equation model.

From this perspective, System Dynamics models and differential equation modeling are one and the same. A System Dynamics model can be expressed using differential equation notation and vice versa. To see this in more detail, we can look at the mapping between System Dynamics and differential equation models. There is a one a one-to-one direct correspondence between the key System Dynamics primitives and components of a differential equation model.

[1] Specifically those where the denominator in the derivative dX/dt is always dt: a very wide class of commonly used models.

System Dynamics Primitive	Differential Equation Equivalent
Stock	State Variable (X, Y, etc...)
Flow	Derivative (dX/dt, dY/dt, etc...)
Variable	Constants/Parameters (α, β, etc...)

Since they do not differ significantly from a mathematical standpoint, what separates these two approaches to modeling? Where System Dynamics and differential equation modeling differ is in their focus and philosophy. The primary goal for differential equation modelers is analytic tractability (in other words, how easy is it to mathematically manipulate and understand the model's equations). This analytic tractability allows these modelers to derive definite results and conclusions from the model's equations. System Dynamics modelers generally are less concerned about analytic tractability and are more comfortable with simulating the model and drawing conclusions from observed trajectories and numerical results.

System Dynamics modelers, to go further, care greatly about communicating their models, deliberately mirroring reality to some extent and exploring the consequences of feedback. The differing focuses on communication between System Dynamics modelers and differential equation modelers can be seen in the method of naming variables. Differential equation models are generally dominated by abstract Greek symbols (e.g. α) while System Dynamics models generally clearly spell out variable names (e.g. "Birth Rate") and additionally use a model diagram to illustrate and communicate the relationships between different parts of the model.

Solving Differential Equations

Given a differential equation or System Dynamics model specification, how do you go about determining the results of the model? This is typically referred to as "solving" the model. Since differential equation models and system dynamics models are essentially one and the same, the techniques used to solve differential equations can be directly applied to System Dynamics models and they are the techniques used by Insight Maker when you simulate any of the models in this book.

For most of the rest of this chapter, we use the differential equation terminology instead of the System Dynamics one. We do so first because it is more concise and more elegantly addresses the issues discussed in this chapter, but also because we want to familiarize you with its terminology and concepts. If you ever get lost, just refer to the System Dynamics to differential equation translation table we showed above.

Let's start our discussion of solving differential equations using our simple population model. As you recall, this model was:

$$\begin{aligned} P(0) &= 100 \\ \frac{dP}{dt} &= \alpha \times P \end{aligned}$$

What is the size of the population, at, let's say $t = 10$ given an α of 0.1? Calculus can be used to solve the model and answer this question. First we separate the terms of the derivative and integrate both sides of the equation. Thereafter it is a simple matter of algebra to solve for P:

$$\begin{aligned} \frac{dP}{dt} &= \alpha \times P \\ dP &= \alpha \times P \; dt \\ \frac{1}{P} \; dP &= \alpha \; dt \\ \log(P) &= \alpha \times t + A \\ P &= e^{\alpha \times t + A} \\ P &= B \times e^{\alpha \times t} \end{aligned}$$

In this equation two new variables A and B appeared (where we arbitrarily set $B = e^A$). These are unknown integration constants[2]. We can determine the values of the integration constants based on the initial conditions of the model, as we specified earlier that $P(0) = 100$. We evaluate the solution of the model at this initial condition to determine the value of B.

$$\begin{aligned} P &= B \times e^{\alpha \times t} \\ 100 &= B \times e^{\alpha \times 0} \\ 100 &= B \end{aligned}$$

Thus our generic equation for P at any time and for any α is:

$$P = 100 \times e^{\alpha \times t}$$

Plugging in $\alpha = 0.1$ and $t = 10$, we obtain:

$$\begin{aligned} P &= 100 \times e^{0.1 \times 10} \\ &= 271.828... \end{aligned}$$

[2]Recall from calculus that if A is a constant, then $x^2 + A \; dx = 2 \times x$. When we integrate $2 \times x$ we need to add back in the constant term. We don't know the value of this constant term immediately and we have to determine it later on.

For this simple population model we have shown that we can obtain the precise population value at any point in the future. It took a fair amount of algebra even for such a simple model, but we did it!

Unfortunately, many differential equation models cannot be solved using these techniques. For most complex models in practice, it is impossible to analytically determine the values of the state variables in the future. This inability to solve a model can be true for even very simple models. Take for example the following growth model similar to our original one:

$$P(0) = 100$$
$$\frac{dP}{dt} = \alpha \times P \times \log(P)$$

We have simply added a logarithm of P into our growth rate. Despite the smallness of this change, this model is now impossible to solve analytically. There is no analytic solution possible, but feel free to give it a try yourself (but please don't try too hard; we promise there is no solution). When developing complex models it should generally to be assumed that in practice no analytical solution will be available. In cases like these, how can we go about developing solutions to the equations and determining the trajectory of the state variables in the system?

The answer is numerical approximation. Even if we can't solve the model equations analytically, we will always be able to approximate their results numerically. A number of different algorithms exist that allow us to approximate the solution to differential equations by repeatedly plugging values into them. To discuss these methods, it is useful to introduce some additional mathematical notation.

In our previous models, we have only looked at systems with a single state variable at a time. However, we can also consider systems containing multiple state variables. The Lotka-Volterra predator prey system we looked at earlier in the book is an example of this. Given two populations of animals – let's say a population of wolves (W) and a population of moose (M) – where the first population preys upon the second, we obtain a paired set of differential equations representing this predator prey relationship:

$$\frac{dM}{dt} = \alpha \times M - \beta \times M \times W$$
$$\frac{dW}{dt} = \gamma \times M \times W - \delta \times W$$

When looking at algorithms to solve sets of equations like these numerically, it can be useful to denote $\mathbf{y}$ as a vector of all the state variables in the model. So for the case of the exponential growth model $\mathbf{y} = [P]$ while for the Lotka-Volterra

model $\mathbf{y} = [M, W]$. When using this notation, $\mathbf{y_t}$ indicates the vector of state variable values at a specific point in time, so $\mathbf{y_0}$ are the initial conditions for this model.

Additionally, we can denote $\mathbf{y}'$ as the vector of derivatives for the different state variables. We treat these derivatives as functions of the current time and the values of the other state variables. So, for instance, to determine the rate of change of the state variables in a model at $t = 10$, we would write $\mathbf{y}'(\mathbf{y_{10}}, 10)$ where $\mathbf{y_{10}}$ are the values of the state variables at $t = 10$.

The use of this notation might seem cumbersome, but it allows us to elegantly describe the mathematics of numerical solution algorithms without getting tied up in the details of a specific model.

Euler's Method

Leonhard Euler

The most basic numerical solution algorithm for differential equations is Euler's method[3]. Simply put, assuming we know the state of the system at time t and we wish to estimate the state of the system at time $t + \Delta t$ (where Δt is pronounced "delta-t" and represents the change in time) we can use the following equation:

$$\mathbf{y_{t+\Delta t}} = \mathbf{y_t} + \Delta t \times \mathbf{y}'(\mathbf{y_t}, t)$$

Let's walk through what this equation is doing. It first takes the derivatives for the state variables at the current point in time. It multiplies these rates of

[3]Leonhard Euler was a brilliant 18th century Swiss mathematician who made many great advances in the theoretical and applied mathematics.

change by the Δt (how far in the future we want to know the results) and adds this change to the values of the state variables at the starting point in time. The result is an estimate of what the values in the future should be.

Let's now apply this to a concrete example. Start with our population scenario, but instead of exponential growth we have a fixed inflow of people at a rate of 20 per year. At $t = 0$ we have 100 people and we want to know the population in 10 years, using Euler's method we obtain the following:

$$\begin{aligned} P_{10} &= P_0 + \Delta t \times \frac{dP}{dt} \\ &= P_0 + 10 \times 20 \\ &= 100 + 200 \\ &= 300 \end{aligned}$$

Thus the population size in 10 years will be 300. In this simple example, Euler's method works perfectly and generates the exact same answer as we would have found using analytic solutions.

In general, however, we won't be so lucky. For most problems Euler's method will generate results that contain some level of error compared to what the true value should be. To see this let's explore our exponential growth model again with an α of 0.1. As a reminder, this model is:

$$\begin{aligned} P(0) &= 100 \\ \frac{dP}{dt} &= 0.1 \times P \end{aligned}$$

As we showed earlier, the precise solution to this model for $t = 10$ (to three decimal places) is 271.828. Let's see what we get using Euler's method with $\Delta t = 10$. Carrying out similar calculations as before we get:

$$\begin{aligned} P_{10} &= P_0 + \Delta t \times \frac{dP}{dt} \\ &= P_0 + 10 \times (0.1 \times P_0) \\ &= 100 + 10 \times (0.1 \times 100) \\ &= 100 + 10 \times 10 \\ &= 100 + 100 \\ &= 200 \end{aligned}$$

So using Euler's method we obtain an estimate 200 for the population size at $t = 10$ when we know the true value should be around 272. That's a pretty large error! Why does this error come about? Why do we so significantly underestimate the final population size?

The reason is that we calculate the population's rate of change only at $t = 0$. For each of the ten years we are simulating, we assume the population grows at the rate it would if there were exactly 100 people. However, the population size is constantly increasing during these ten years, so the rate at which it grows should also be increasing. Imagine, the case of a bank account with an interest rate of 10% yearly. The bank account grows over time so the interest earned should also grow from year to year. It's the same principle of compounding here.

How do we address this issue? Using Euler's method, we can do it simply by changing how often we calculate the rates of change. In our previous calculation, we went straight from $t = 0$ to $t = 10$ all in one step, we used a Δt in Euler's equation of 10. However, we could employ an alternate calculation strategy where, for instance, we stepped from $t = 0$ to $t = 5$, recalculated the derivative based on the new population size and then stepped from $t = 5$ to $t = 10$. This would be equivalent to used a Δt of 5 and iterating the algorithm twice. Here is what we get doing this:

$$\begin{aligned} P_5 &= P_0 + \Delta t \times \frac{dP}{dt} \\ &= P_0 + 5 \times (0.1 \times P_0) \\ &= 100 + 50 \\ &= 150 \\ P_{10} &= P_5 + \Delta t \times \frac{dP}{dt} \\ &= P_5 + 5 \times (0.1 \times P_5) \\ &= 150 + 5 \times 15 \\ &= 150 + 75 \\ &= 225 \end{aligned}$$

That result is certainly better, and we cut our error by over 33%. However, the error is still too large for most practical purposes. To improve the numerical estimation even more, we can apply smaller and smaller Δt's. You probably have a good grasp of the calculations now, so let's just show the results for each step of the simulation. We'll look at $\Delta t = 2$ and $\Delta t = 1$.

t	P
0	100
2	120
4	144
6	172.8

8	207.4
10	248.8

t	P
0	100
1	110
2	121
3	133.1
4	146.4
5	161.1
6	177.2
7	194.9
8	214.4
9	235.8
10	259.4

We see that as Δt gets smaller and smaller our results become more and more accurate. However, they are never perfect. There is always some error. Even if we made Δt as small as 0.1 (requiring 100 simulation steps), our final population size would be calculated to be 270, an error just under 1%.

Figure 1 illustrates the application of Euler's method to numerically estimate the trajectory for an example function. The smaller the Δt's in the estimation are, the better the results will be. Other terms that can be used in place of Δt are "Step Size", "Time Step" or just "DT". We prefer not to use the notation DT as it is easily confusable with the dt from differential equations. The latter indicates an infinitesimally small change, while step sizes are never infinitesimally small.

As you decrease the step size for the simulation, the results of the simulation become more and more accurate[4]. The cost of this increased accuracy, however, is increased computation time. The computation time required by your model

[4]It is important to note at this point that when we discuss accuracies in this context we are specifically referring to models composed of continuous differential equations. If you are using agent based modeling or have discontinuities in your models – which could occur if you use `IfThenElse()` logic – then a smaller step size may not provide additional accuracy when there is some fundamental time step logic to the model.

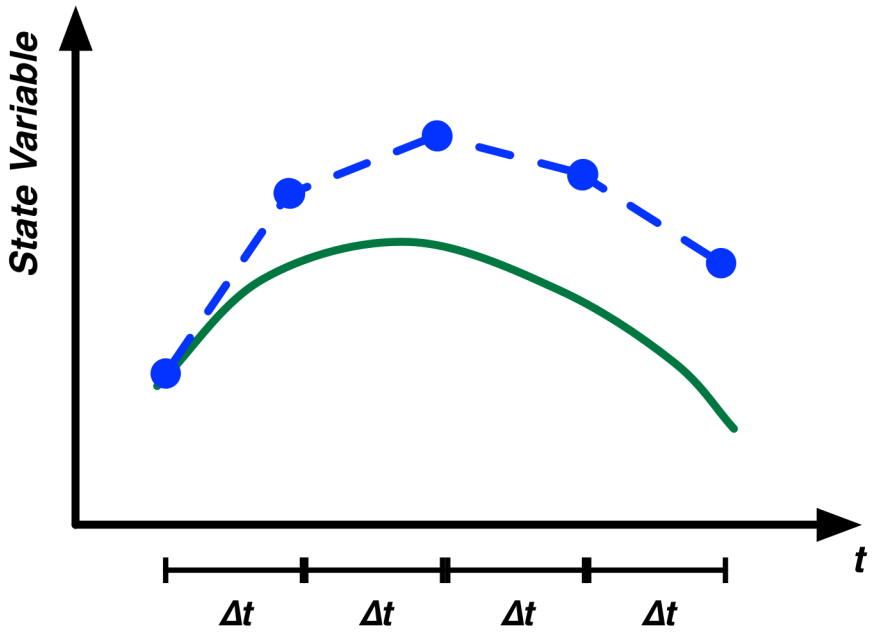


Figure 1. Euler's method at work. The true trajectory for the illustrative state variable is shown in green. Euler's method estimate of this trajectory is shown in blue.

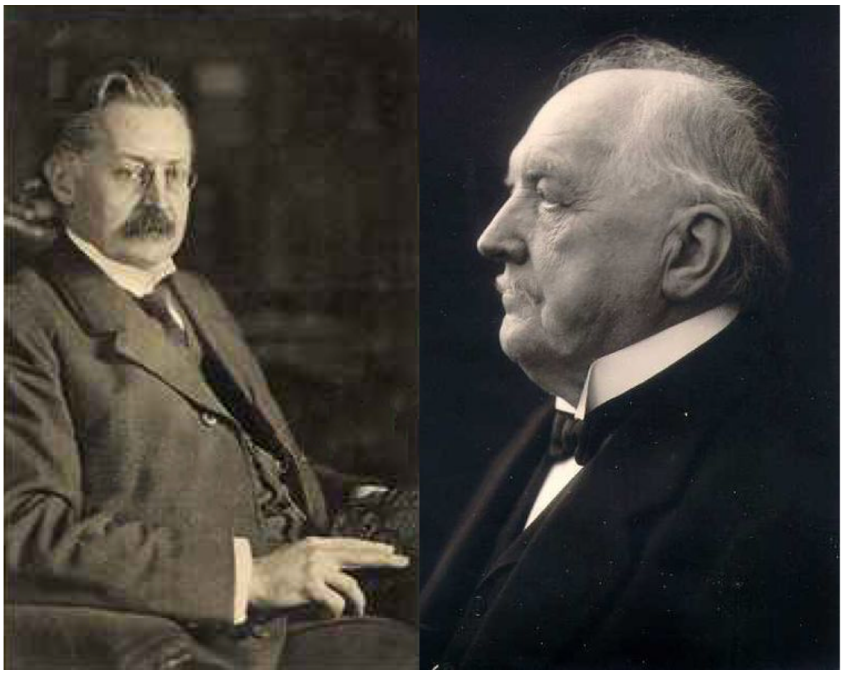
is directly proportional to 1 over the step size. Thus, if you cut the step size in half, your model will take twice as long to complete simulating.

In general, you want a step size small enough that your results are "accurate enough," but one that isn't so small that the simulation takes too long to complete. A rule of thumb for choosing the step size is to choose a starting step size that results in a fast simulation. Then cut the value of the step size in half and simulate the model again. If the results have not change materially between these two simulations, keep the larger step size. If the results have changed, cut the step size in half again and repeat until the results cease to change.

Runge-Kutta Methods

Euler's method is not the only technique that can be used to numerically solve differential equations. Another popular set of techniques are called Runge-Kutta methods. Runge-Kutta methods are a family of numerical differential equation solvers. In fact Euler's method itself can be classified as a simple Runge-Kutta method.

One particular member of the Runge-Kutta family of methods that is widely



Carl Runge and Martin Kutta

used is a 4th-order Runge-Kutta method. This method differs from Euler's method in that for each step, it evaluates the model multiple times and averages the resulting derivatives. Briefly, the driving set of equations for this method is as follows:

$$\mathbf{y_{t+\Delta t}} = \mathbf{y_t} + \Delta t \frac{\mathbf{a} + 2 \times \mathbf{b} + 2 \times \mathbf{c} + \mathbf{d}}{6}$$

Where:

$$\mathbf{a} = \mathbf{y}'(\mathbf{y_t}, t)$$
$$\mathbf{b} = \mathbf{y}'(\mathbf{y_t} + \frac{\Delta t}{2} \times \mathbf{a}, t + \frac{\Delta t}{2})$$
$$\mathbf{c} = \mathbf{y}'(\mathbf{y_t} + \frac{\Delta t}{2} \times \mathbf{b}, t + \frac{\Delta t}{2})$$
$$\mathbf{d} = \mathbf{y}'(\mathbf{y_t} + \Delta t \times \mathbf{c}, t + \Delta t)$$

What this algorithm does is first compute the derivatives of the system at the current time ($\mathbf{a}$) and use them to move the system forward to $t + \Delta t/2$. The derivatives are evaluated at $t + \Delta t/2$ ($\mathbf{b}$) and this new set of derivatives is used to again move the system from t to $t + \Delta t/2$. A third set of derivatives are evaluated again at this mid-point ($\mathbf{c}$) and they are the used to move the system

from t to $t + \Delta t$. A fourth set of derivatives are evaluated at this point (**d**). The system is then returned to its starting point and a weighted average of derivatives are used to move the system the full time step. This weighting puts most of the weight on the middle two derivatives instead of the derivatives from the end points.

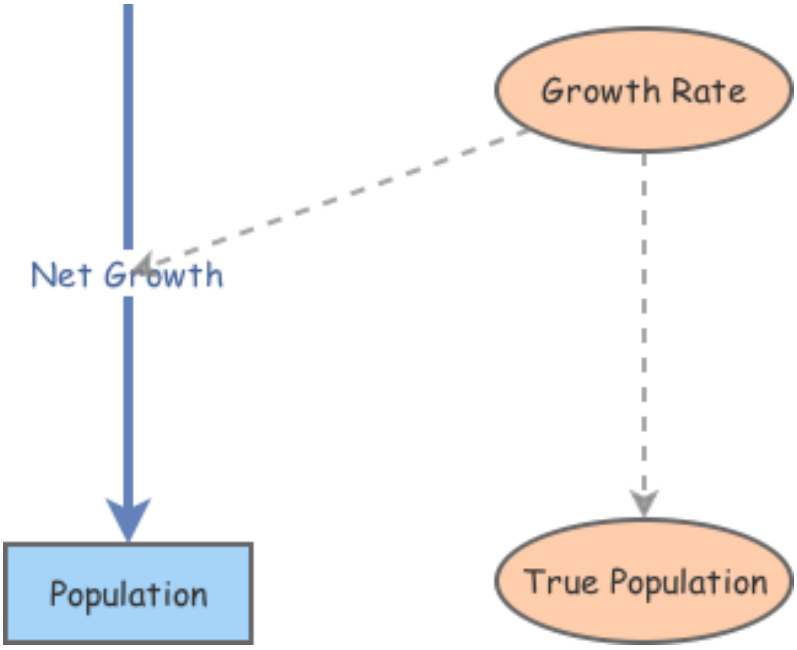
This 4th-order Runge-Kutta method is generally much more accurate than Euler's method for a given step size. Using a step size of 10 for our earlier population model, the Runge-Kutta method generates a value of 270.8. A step size of 5 yields a results of 271.7, just a smidgeon away from the precise value of 271.8. Recall that for Euler's method, even with a step size of 0.1 we still were not as accurate as the Runge-Kutta method with a step size of 5. Now it is true that this 4th-Order Runge-Kutta method does a lot more work than Euler's method for each step. It evaluates the model for times and has to do some averaging of derivatives. However, it is still much more accurate than Euler's method for an equivalent level of computational effort.

Numerical Solution Algorithms

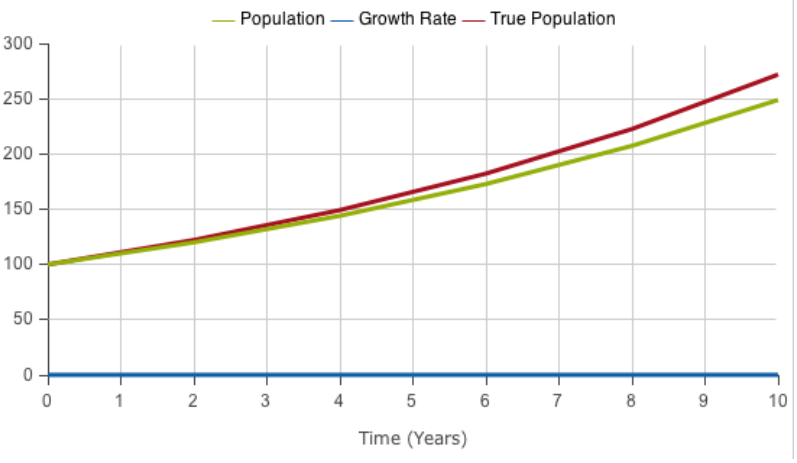
This model explores the selection of the simulation step size and differential equation solution algorithm.

1. Create a new **Stock** named [**Population**].
2. Change the **Initial Value** property of the primitive [**Population**] to `100`.
3. Create a new **Flow** going from empty space to the primitive [**Population**]. Name that flow [**Net Growth**].
4. Create a new **Variable** named [**Growth Rate**].
5. Change the **Equation** property of the primitive [**Growth Rate**] to `0.1`.
6. Create a new **Link** going from the primitive [**Growth Rate**] to the primitive [**Net Growth**].
7. Change the **Flow Rate** property of the primitive [**Net Growth**] to `[Growth Rate]*[Population]`.
8. Create a new **Variable** named [**True Population**].
9. Create a new **Link** going from the primitive [**Growth Rate**] to the primitive [**True Population**].
10. Change the **Equation** property of the primitive [**True Population**] to `100*Exp([Growth Rate]*Years)`.

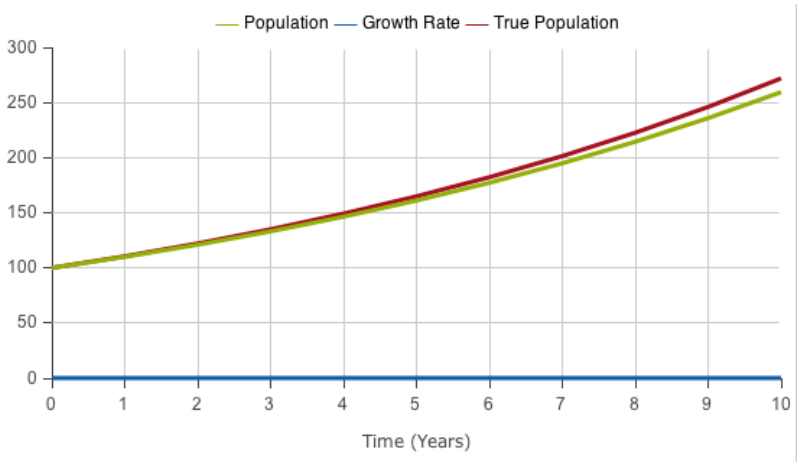
11. Change the **Simulation Length** property of the Time Settings to 10.
12. The model diagram should now look something like this:



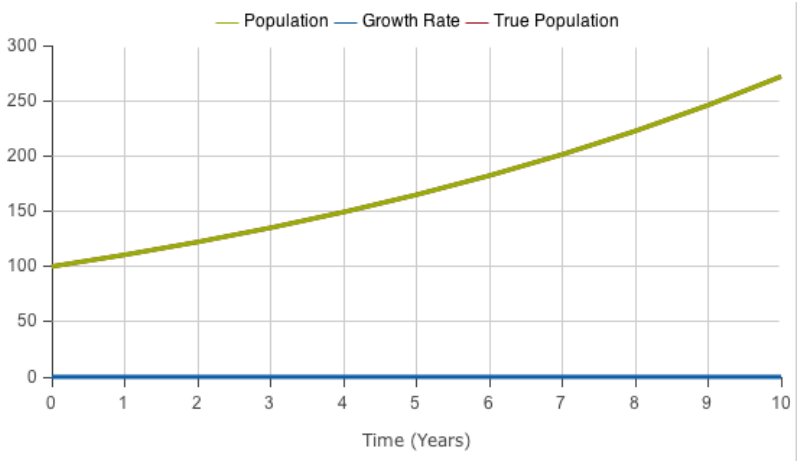
13. Let's now implement the simple exponential growth model we have discussed in this chapter. We have a population that starts with 100 people and increases at a rate of 10
14. First, we'll use Euler's method with a step size of 2 years and simulate the model.
15. Change the **Simulation Time Step** property of the Time Settings to 2.
16. Run the model. Here are sample results:



17. As we can see these results aren't very accurate. The value of the numerical estimated [**Population**] is quite different from the analytically determined value in [**True Population**]. Let's reduce the step size to 1 year and try again.
18. Change the **Simulation Time Step** property of the Time Settings to `1`.
19. Run the model. Here are sample results:



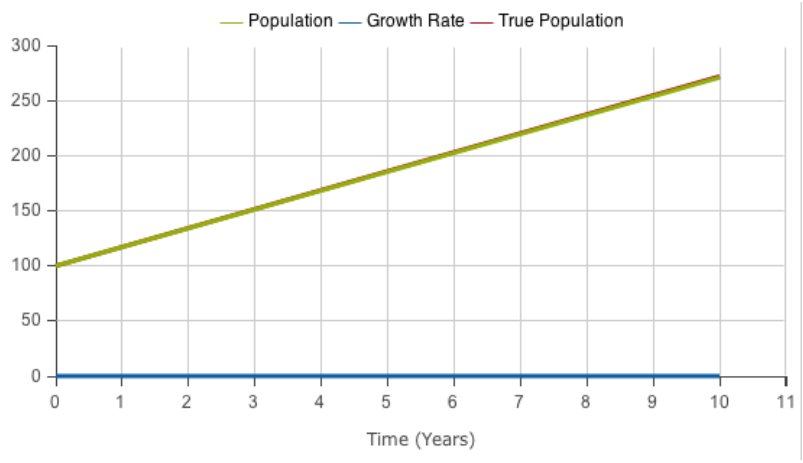
20. This is better, but we're still off by a fair amount. We could experiment with continuing to reduce the step size, but let's instead switch now to the more accurate Runge-Kutta method. Will simulate the model again with a step size of 1 using the 4th-Order Runge-Kutta solution algorithm.
21. Change the **Analysis Algorithm** property of the Time Settings to `RK4`.
22. Run the model. Here are sample results:



23. That's a lot better! It's so close to being perfect that we can't even see the difference between the two lines in the figure. Just to be clear, let's see how quickly the results degrade when we increase the step size. Let's set the step size to 10 and simulate the model again.

24. Change the **Simulation Time Step** property of the Time Settings to `10`.

25. Run the model. Here are sample results:



26. That's still very good and much better than Euler's Method with a step size of 1. Why don't you go ahead now and experiment with different step sizes and the two solution methods to get a feel for their accuracies.

Other Solution Techniques

While being a brief introduction into numerical solution methods for differential equations, this should provide you with the background you need to intelligently make decisions about controlling the simulation of your models. It should help you identify potential sources of errors in your model and help you to adjust you simulation configuration to account for them.

The two methods we have looked at for solving differential equation models – Euler's method and a 4th-Order Runge-Kutta method – are widely used and they are what are built into Insight Maker. In addition to these two techniques, however, there are many other methods that are used in practice and you should be aware of this richer ecosystem of solution techniques.

Although we do not have space here to delve into the full ecosystem of numerical differential equation algorithms, it is useful to discuss one variant briefly: the

adaptive step size algorithm. The methods we have looked at here use a fixed step size specified at the beginning of a simulation. Many models, however, might be characterized by highly variable trajectories. Part of the trajectory might be very smooth and unchanging while other parts might experience numerous rapid changes.

When using a fixed step size algorithm like the ones illustrated above, the step size must be set for the worse case scenario. The step size must be set to a small enough value to account for the rapidly changing areas. That said, the precision of this small step size is unnecessary on the smooth regions of the trajectory where the algorithm must do extra work for minimal gain in precision. Ideally, we would want to have a small step size for the rapidly changing areas and a large one for the smooth regions. This would result in the best of both worlds: high accuracy and quick computation.

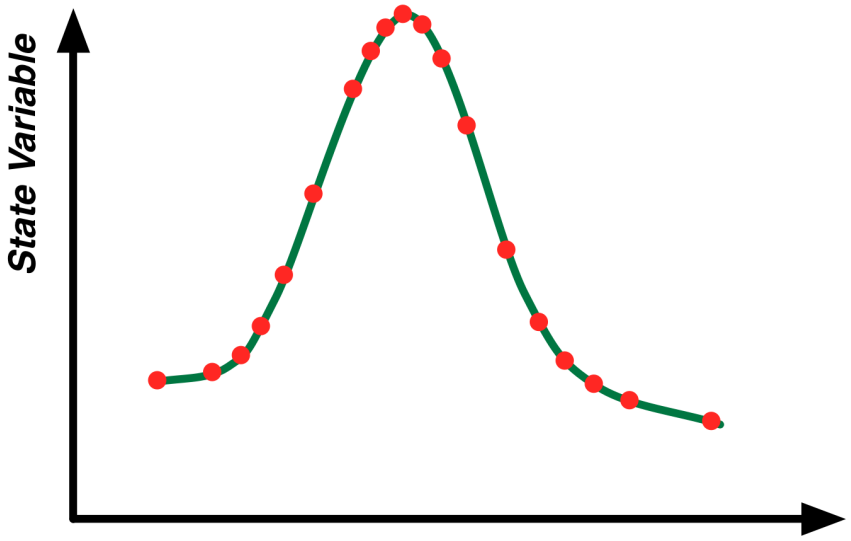


Figure 2. Illustration of an adaptive step size algorithm. Dots show the location of model evaluations. Evaluations are clustered around changes in the derivatives.

Adaptive step size algorithms do just that. They adjust the step size dynamically based on the behavior of the model's derivatives. If the derivatives change rapidly, then the step size will be automatically shrunk; if the derivatives are constant or change very slowly the step size will automatically grow. Figure 2 illustrates the location of steps for an illustrative model using an adaptive step size algorithm. The steps are clustered around changes in the trajectory's derivatives in an attempt to maximize predictive accuracy while minimizing computation effort.

Analyzing Differential Equations

Although the trajectory for the state variables in differential equation models generally cannot be determined analytically, several key properties of models can often still be determined. These properties include:

- The location of equilibrium points
- The stability of the equilibrium points

An equilibrium point is defined as a set of state variable values that will cause the system to cease to change. Once the system enters an equilibrium configuration, it will not leave that configuration without an external stimulus. For instance, in our exponential growth model a single equilibrium point exists: that of zero people. If the population is empty, then the population will not grow and instead remain at 0 indefinitely.

In the exponential growth population model there is only one equilibrium point ($P = 0$). In other models you may have multiple equilibrium points. In a model of a highly infectious, incurable disease you can imagine a system where two equilibrium points exist: one where no one is infected and a second point where everyone is infected. As long as there were no infectious individuals, the population would remain healthy. If just a single infected individual were introduced into the population, the infection would, however, spread until everyone was infected and the population would then remain at that point (remember this hypothetical disease is incurable).

Multiple types of equilibria exist. Figure 3 illustrates what is known as the *stability* of equilibrium points. Each of the three panes in this figure show a different form of equilibrium for the ball. In all three the balls are in equilibrium: if the no external forces come into play, the balls will not move. What differs in each of the three is what occurs if the balls are displaced a small amount.

Stable Equilibrium : In this type of equilibrium the ball will return to its original position if it is displaced. The structure of the system is such that the system is naturally attracted to the point of equilibrium. To use the physical metaphor, the equilibrium is at the bottom of a dip and the system naturally rolls into it.

Unstable Equilibrium : Here the ball will move further and further away from the point of equilibrium if it is displaced even a small amount. The equilibrium is unstable in that if we are just a small distance away from it, we move further away from it. To use the physical metaphor, the equilibrium is at the top of the hill and the system will move away from it unless it is placed at the exact point of equilibrium.

"Neutrally Stable Equilibrium" : This is a less common form of equilibrium and goes by several different names. In this case if the ball is moved it will stay fixed at its new location. It will not move closer to or further from the

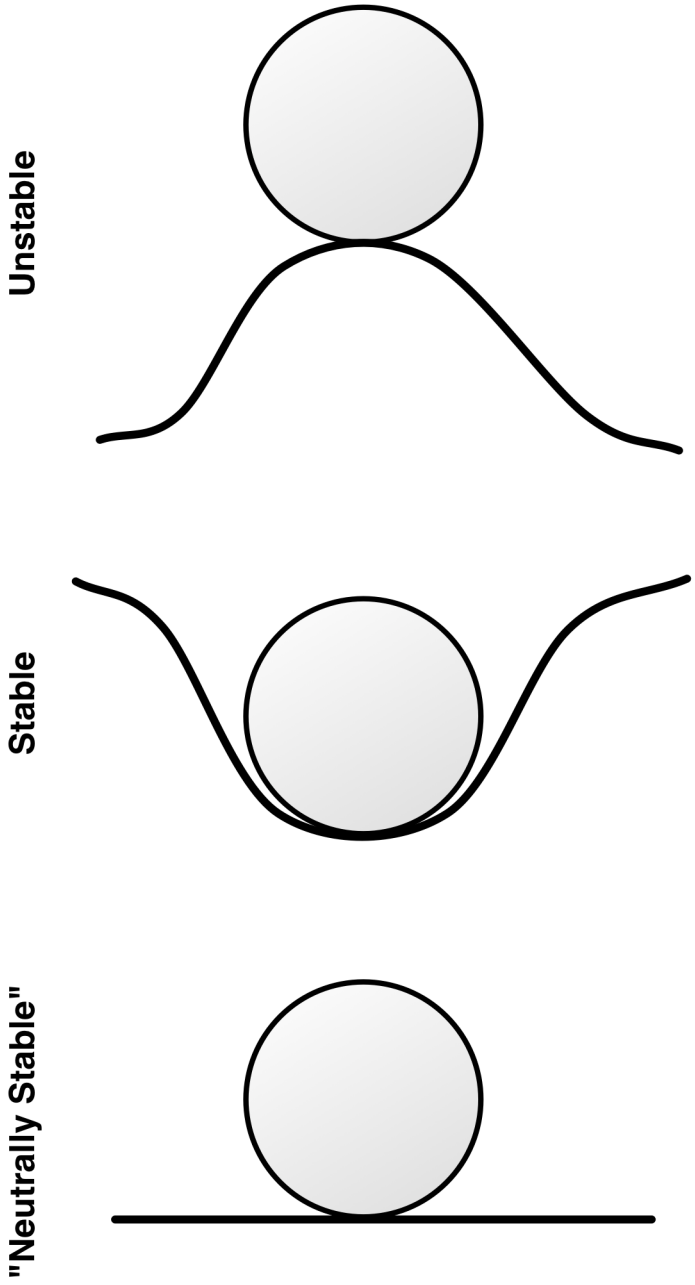


Figure 3. Three different types of stability.

original equilibrium. Of the three types of equilibrium, this one is less interest or relevance in practice.

In the case of the highly infectious disease model, an equilibrium of everyone being healthy would be classified as an unstable equilibrium. The equilibrium would persist as long as no one brought the disease into the population (someone would not just spontaneously become ill), but if as little as a single sick person entered the population, the population would move further and further away from the equilibrium point of everyone being healthy and would never naturally return to it.

The equilibrium point of everyone being sick is, on the other hand, a stable equilibrium as no one recovers from the disease on their own. Even if you introduced healthy people into a population of sick individuals – moving the population away from the equilibrium – they too will eventually become sick restoring the population to the equilibrium of everyone being sick.

Equilibrium Points

Often, we can determine the equilibrium points for a system without fully needing to solve the trajectory for the state variables. Let's implement the simple disease model we've been discussing. We'll do so for both a differential equation model and a System Dynamics model, but we'll rely on differential equation version to do our analytic analysis.

One way to express the differential version of the model is to define two state variables: the number of healthy people (H) and the number of sick people (S). The rate of infection between sick and healthy people can be made a function of the number of people in each category. Clearly, if there are no sick people the infection rate is 0; but, just as clearly, if everyone is already sick then the infection rate will also be zero. One workable differential equation model to implement this behavior is shown below:

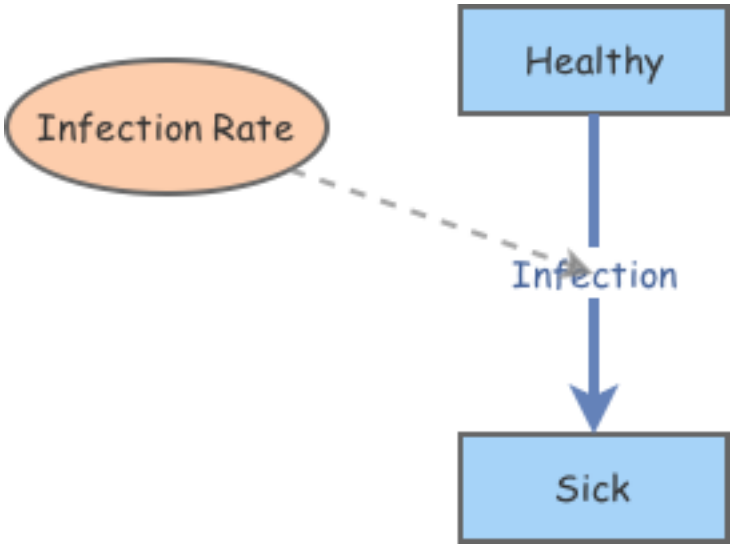
$$\begin{aligned} H(0) &= 100 \\ S(0) &= 1 \\ \frac{dH}{dt} &= -\alpha \times H \times S \\ \frac{dS}{dt} &= \alpha \times H \times S \end{aligned}$$

This model uses a single parameter (α) to control the infection rate. *alpha* is a non-zero positive value; the smaller α is, the slower the infection will progress and vice versa. This notation illustrates one of the clumsier aspects of implementing stock and flow models using differential equations. The flow values between two stocks have to be repeated twice once for each of the two connected state variable's derivatives.

Incurable Disease

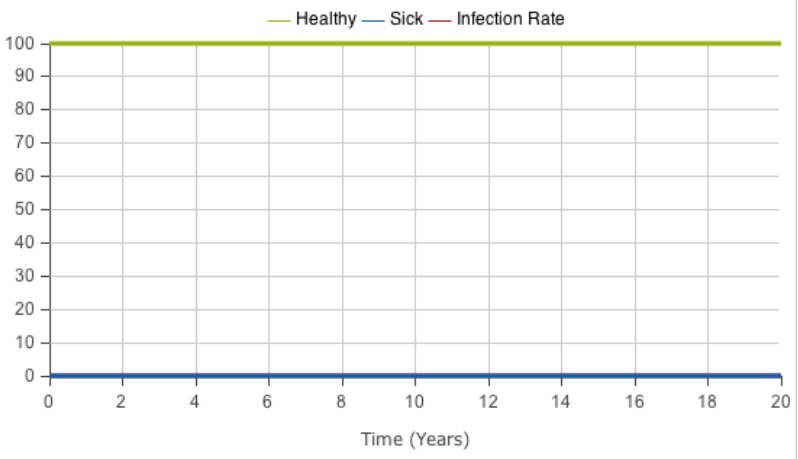
This model illustrates stable and unstable equilibria using the scenario of an incurable disease in a population.

1. Create a new **Stock** named [**Healthy**].
2. Create a new **Stock** named [**Sick**].
3. Create a new **Flow** going from the primitive [**Healthy**] to the primitive [**Sick**]. Name that flow [**Infection**].
4. Create a new **Variable** named [**Infection Rate**].
5. Create a new **Link** going from the primitive [**Infection Rate**] to the primitive [**Infection**].
6. The model diagram should now look something like this:

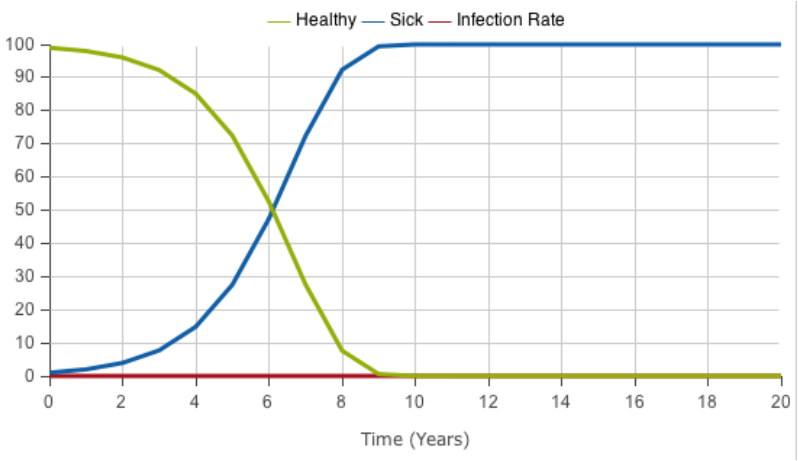


7. This is the structure of our model. We have two stocks of people with people moving from the healthy stock to the sick stock as they become infected. Let's add the values and equations now.
8. Change the **Initial Value** property of the primitive [**Healthy**] to `100`.
9. Change the **Initial Value** property of the primitive [**Sick**] to `0`.
10. Change the **Equation** property of the primitive [**Infection Rate**] to `0.01`.

11. Change the **Flow Rate** property of the primitive [**Infection**] to `[Infection Rate]*[Healthy]*[Sick]`.
12. There our model is fully setup. We've set it to start with everyone being healthy.
13. Run the model. Here are sample results:



14. These results are quite stable. Everyone is healthy and no one gets sick. That indicates we have an equilibrium here. Let's now experiment by making a single person in the population sick.
15. Change the **Initial Value** property of the primitive [**Sick**] to `1`.
16. Change the **Initial Value** property of the primitive [**Healthy**] to `99`.
17. Run the model. Here are sample results:



18. That's more interesting! We can see that everyone being healthy is an unstable equilibrium as the system moves away from it if we deviate from it by even a small amount. We can also see that the second equilibrium (everyone being sick) is stable as the system moves towards it naturally.

Finding the equilibria for differential equation models is by-and-large straightforward analytically. We simply need to harness the definition of an equilibrium point: an equilibrium point is one where the state variables are constant and unchanging. Since the derivatives represent changes in the state variables, this statement is equivalent to saying the derivatives for the model are 0 at equilibrium points.

Based on this, in order to find the equilibrium points we simply need to set the derivatives in our model to 0 and solve the resulting equations. For the disease model we get:

$$\begin{aligned} H(0) &= 99 \\ S(0) &= 1 \\ 0 &= -\alpha \times H \times S \\ 0 &= \alpha \times H \times S \end{aligned}$$

The initial conditions will determine what equilibrium is arrived at but they do not affect the existence of the equilibria. Furthermore, the two equations we have set to 0 are equivalent[5] so we can simplify these equations to simply be:

$$0 = \alpha \times H \times S$$

Simple inspection reveals that this equation is true if and only if either $H = 0$, $S = 0$, or $\alpha = 0$. Thus we have mathematically shown that our equilibria are either when everyone is sick or everyone is healthy (or there is no infection whatsoever). Granted this is a trivial conclusion for this model and we stated it earlier. However, for more complex models this type of analysis can be very useful and will often reveal that equilibria are functions of the different parameter values in the model and they may enable you to explicitly determine how the equilibria changes as the model configuration changes.

Let's try a more complex example. Remember the predator prey model from earlier? We had the following set of equations to simulate the relationship between a moose and wolf population:

[5] Although we expressed this model as a function of two state variables H and S, it only has one independent state variable. Given the fixed population size, you know the value of H given S and vice versa.

$$\frac{dM}{dt} = \alpha \times M - \beta \times M \times W$$

$$\frac{dW}{dt} = \gamma \times M \times W - \delta \times W$$

Let's determine what the equilibrium values are for this model. As before, we start by setting the derivatives to 0:

$$0 = \alpha \times M - \beta \times M \times W$$

$$0 = \gamma \times M \times W - \delta \times W$$

Solving this set of equations is more difficult than for the disease model. However a little bit of algebra reveals two solutions. One when $M = 0$ and $W = 0$ (there are no animals at all), and the second when $M = \delta/\gamma$ and $W = \alpha/\beta$. This is an example of where the equilibrium location depends on the values of the model parameters.

The Phase Plane

When looking at model results, up until now we have been focused on time series plots and we have mainly been interested in the trajectory of the variables and stocks over time. For the mathematical analysis of differential equations, however, the primary graphical tool is not this time series plot; instead it is what is known as a phase plane plot.

Phase planes are almost like scatterplots. They show one of the state variables plotted against another of the state variables. A scatterplot could be used to show the path for these two variables over the course of a simulation. In the predator prey model the results of a scatterplot of the wolf and moose population will be an ellipsoid. The two populations will cycle continuously. A phase plane plot is similar to this, but rather than just showing one of these cycles for a given simulation run, the phase plane shows the trajectories for *all* combinations of moose and wolf population sizes.

Figure 4 illustrates a phase plane plot for the predator prey system. The trajectory for one set of parameter and state variable values is highlighted in red and, as expected, we see a continual oscillation. We can also see the trajectories for all the other combinations of state variables. We see that the system will always oscillate and the size of this oscillation depends on the initial conditions for the state variables. This illustration provides us with a good deal of information in a single graphic and the phase plane plot is a great way to summarize the behavior of a system with two state variables.

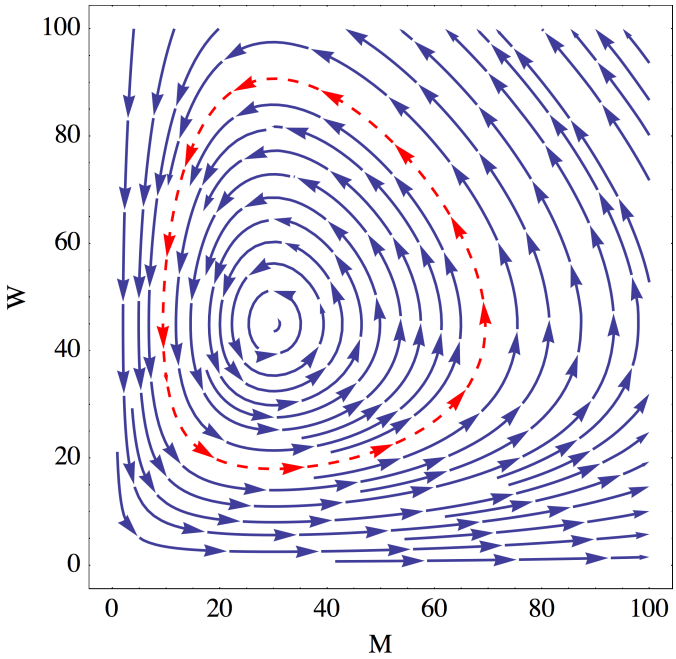


Figure 4. Predator prey phase plane plot. The trajectory for a single set of initial conditions is highlighted in red.

Let's quickly explore the phase plane plots for a simpler system than our predator prey model. Take a system consisting of two state variables[6] both of which grow (or decay) exponentially. These state variables will be assumed to be independent from each other so the value of one does not affect the value of the other:

$$\frac{dX}{dt} = \alpha \times X$$
$$\frac{dY}{dt} = \beta \times Y$$

Clearly, there is an equilibrium point for this model at $X = 0$ and $Y = 0$. There are four general types of behavior around this equilibrium. One when $\alpha > 0$ and $\beta > 0$, one when $\alpha < 0$ and $\beta > 0$, one when $\alpha > 0$ and $\beta < 0$, and one when $\alpha < 0$ and $\beta < 0$. The phase planes for each of the four cases are shown in Figure 5.

[6]Just a helpful reminder if you are starting to get lost in some of this differential equation jargon. A "state variable" is just a stock. Return to the table at the beginning of this chapter to see how these terms relate to the system dynamics modeling terminology we have already learned.

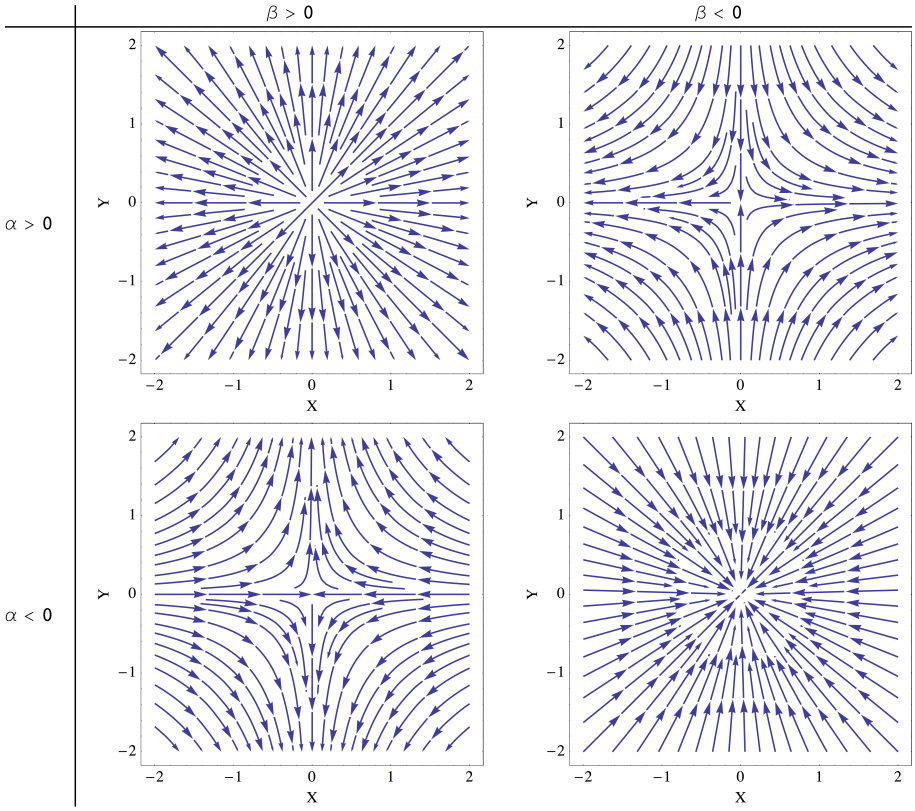


Figure 5. Phase planes for a simple two state variable exponential growth model.

From these plots we can visually determine how the stability of the equilibrium point at $X = 0, Y = 0$ changes as we change α and β. When $\alpha < 0$ and $\beta < 0$, we have a stable equilibrium; in all other cases we have an unstable equilibrium.

Stability Analysis

Now that we have learned how to analytically determine the location of equilibrium points, we may want to determine what type of stability occurs at these equilibria. As we stated earlier, for the incurable disease model it is trivial to arrive at the conclusion that the state of everyone being healthy is unstable while the state of everyone being sick is stable. In more complex models, it may be harder to draw conclusions or the stability of an equilibrium point may change as a function of the model's parameter values. Fortunately, there is a general way to determine the precise stability nature of the equilibrium points analytically.

The procedure to do this is relatively straightforward, but the theory behind it

can be difficult to understand. The first key principle that must be understood is that of "linearization". To get a feel for linearization, let's take the curve in Figure 6. Clearly this curve is not linear. It has lots of bends and does not look at all like a line.

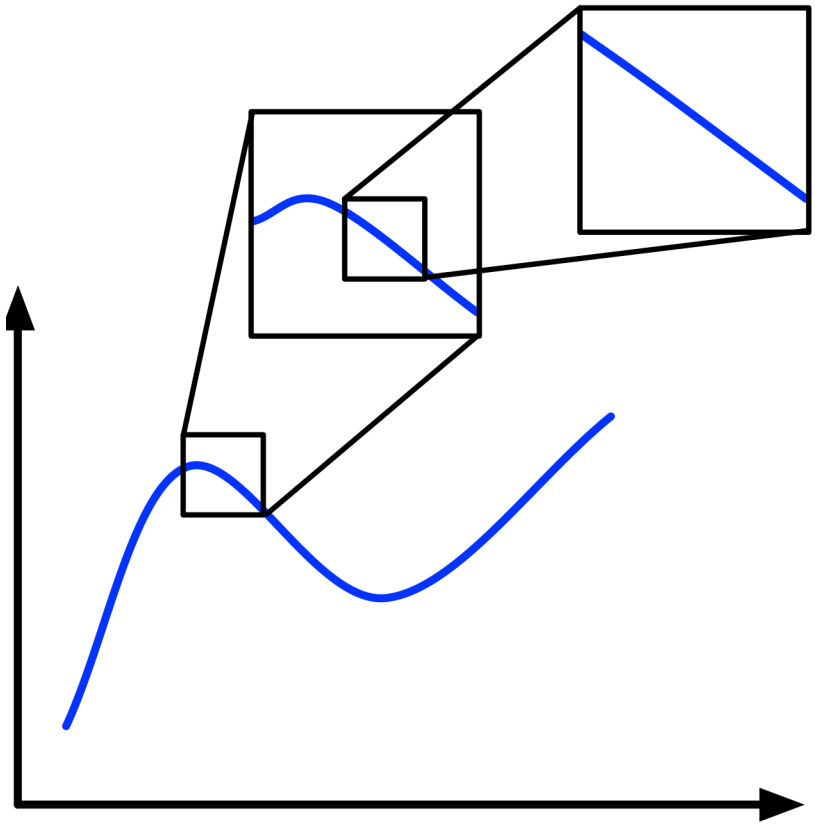


Figure 6. As we zoom in on a function it becomes more and more linear.

If we zoom in on any one part of the curve, however, the section we are zoomed in on starts to straighten out. If we keep zooming in, we will eventually reach a point where the section we are zoomed in on is effectively linear: basically a straight line. This is true for whatever part of the curve we zoom in on[7]. The more bendy parts of the curve will just take more zooming to convert them to a line.

We can conceptually do the same process for the equilibrium points in our phase planes. Even if the trajectories of the state variables in the phase planes are

[7]The one exception to this rule is if your curve is some sort of fractal. In this case no matter how much you zoom in on it, it will never become straight. In practice, however, this caveat is a non-issue.

very curvy, if we zoom in enough on the equilibrium points, the trajectories at a point will eventually become effectively linear. The simple, two-state variable exponential growth model we illustrated with phase planes above are examples of a fully linear model. If we zoom in sufficiently on the equilibrium points for most models, the phase planes for the zoomed-in version of the model will eventually start to look like one of these linear cases.

Mathematically, we apply linearization to an arbitrary model by first calculating what is called the Jacobian matrix of the model. The Jacobian matrix is the matrix of partial derivatives of each of derivatives in the model with respect to each of the state variables:

$$\text{Jacobian} = \begin{bmatrix} \frac{\partial}{\partial X}X' & \cdots & \frac{\partial}{\partial Z}X' \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial X}Z' & \cdots & \frac{\partial}{\partial Z}Z' \end{bmatrix}$$

The Jacobian is a linear approximation of our (potentially) non-linear model derivatives. Let's take the Jacobian matrix for the simple exponential growth model:

$$\frac{dX}{dt} = \alpha \times X$$
$$\frac{dY}{dt} = \beta \times Y$$

$$\text{Jacobian} = \begin{bmatrix} \frac{\partial}{\partial X}\alpha \times X & \frac{\partial}{\partial Y}\alpha \times X \\ \frac{\partial}{\partial X}\beta \times Y & \frac{\partial}{\partial Y}\beta \times Y \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix}$$

This is complicated so don't worry if you don't completely understand it! Once you have the Jacobian, you calculate what are known as the eigenvalues of the Jacobian at the equilibrium points. This is also a bit complicated, so if your head is starting to spin, just skip forward in this chapter!

Nonetheless, eigenvalues and their sibling eigenvectors are an interesting subject. Given a square matrix (a matrix where the number of rows in equals the number of columns), an eigenvector is a vector which, when multiplied by the matrix, results is the original vector multiplied by some factor. This factor is known as an eigenvalue as is usually denoted λ. Given a matrix $\mathbf{A}$, an eigenvalue λ with associated eigenvector $\mathbf{V}$; the following equation will be true:

$$\mathbf{A} \times \mathbf{V} = \lambda \times \mathbf{V}$$

Let's look at an example for a 2×2 matrix:

$$\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \times \mathbf{V} = \lambda \times \mathbf{V}$$

What eigenvector and eigenvalue combinations satisfy this equation? It turns out there are two key ones:

$$\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 2 \\ 1 \end{bmatrix} = 2 \times \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} -1 \\ 1 \end{bmatrix} = -1 \times \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Naturally, any multiple of an eigenvector will also be an eigenvector. For instance, in the case above, $[1, 0.5]$ and $[-2, 2]$ are also eigenvectors of the matrix.

We can interpret eigenvectors geometrically. Looking at the 2×2 matrix case, we can think of a vector as representing a coordinate in a two-dimensional plane: $[x, y]$. When we multiply our 2×2 matrix by the point, we transform the point into another point also in the two-dimensional plane. Due to the properties of eigenvectors, we know that when we transform an eigenvector, the transformed point will just be a multiple of the original point. Thus when a point that is on a matrix's eigenvector is transformed by that matrix, it will move inwards or outwards from the origin along the line defined by the matrix's eigenvector.

We can now relate the concept of eigenvalues and eigenvectors to our differential equation models. Take a look back at the phase planes for the exponential model example. For each of the phase planes, there are at least two straight lines of trajectories. In these cases the x- axis and the y-axis are the locations of these trajectories. If you have a system on the x- or y-axis in this example it will remain on that axis as it changes. This indicates that for this model, the eigenvectors are the two axises as a system on either of them does not change direction as it develops. That's the definition of of an eigenvector.

For our purposes though, we do not really care about the actual direction or angle for these eigenvectors. We instead care about whether the state variables move inwards or outwards along these vectors. We can determine this from the eigenvalues of the Jacobian matrix. If the eigenvalue for an eigenvector is negative, then the values move inwards along that eigenvector; while if the eigenvalue is positive, they move outward along the eigenvector.

These eigenvalues tell us all we need to know about the stability of the system. Returning to our illustration of stability as a ball on a hill, we can think of these eigenvalues as being the slopes of the hill around the equilibrium point. If the eigenvalues are negative, the ground slopes down towards the equilibrium point forming a cup (leading to a stable equilibrium). If the eigenvalues are positive,

the ground slopes away from the equilibrium point creating a hill (leading to an unstable equilibrium).

Eigenvalues can be calculated straightforwardly for a given Jacobian matrix. Briefly, for the Jacobian matrix J, the eigenvalues λ are the values that satisfy the following equation where det is the matrix determinant and I is the identity matrix.

$$0 = det(J - \lambda \times I)$$

We can do a quick example of calculating the eigenvalues for the Jacobian matrix we derived for our two-state variable exponential growth model.

$$\begin{aligned} 0 &= det\left(\begin{bmatrix}\alpha & 0\\ 0 & \beta\end{bmatrix} - \lambda \times \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right) \\ &= det\left(\begin{bmatrix}\alpha - \lambda & 0\\ 0 & \beta - \lambda\end{bmatrix}\right) \\ &= (\alpha - \lambda) \times (\beta - \lambda) - 0 \times 0 \\ \lambda &= \alpha, \lambda = \beta \end{aligned}$$

That is a fair amount of work to do. It's even more complicated if you have more than two state variables. However, once you have gone through the calculations and determined the linearized eigenvalues for your equilibrium points, you know everything you might want to know about the stability of the system.

In the exponential growth model we can see that when the eigenvalues are both negative we have a stable equilibrium (refer to the graphs we developed earlier), while if either one is positive (or they both are) we have an unstable equilibrium. This makes a lot of sense as if either one is positive it pushes the system away from the equilibrium making it unstable. While if they are both negative then they both push the system towards the equilibrium point. Visualize the ball sitting in the cup or on the hill.

Let's now look at some more examples.

First let's take our simple disease model from earlier. If you recall that model was:

$$\begin{aligned} \frac{dH}{dt} &= -\alpha \times H \times S \\ \frac{dS}{dt} &= \alpha \times H \times S \end{aligned}$$

First let's calculate the Jacobian for this model. We take the partial derivatives of each of the two derivatives with respect to each of the two state variables to create a two-by-two matrix:

$$\text{Jacobian} = \begin{bmatrix} \frac{\partial}{\partial H} - \alpha \times H \times S & \frac{\partial}{\partial S} - \alpha \times H \times S \\ \frac{\partial}{\partial H}\alpha \times H \times S & \frac{\partial}{\partial S}\alpha \times H \times S \end{bmatrix} = \begin{bmatrix} -\alpha \times S & -\alpha \times H \\ \alpha \times S & \alpha \times H \end{bmatrix}$$

Next, we evaluate this Jacobian at one of our equilibrium points. Let's choose the one where the $S = 0$ (no one is sick) and $H = P$ (where P is the population size) so everyone is healthy:

$$\begin{bmatrix} 0 & -\alpha \times P \\ 0 & \alpha \times P \end{bmatrix}$$

We can now find the eigenvalues for this matrix. Once we go through the math we get two eigenvalues: 0 and $\alpha \times P$. What do these mean? Well, since one of the eigenvalues is positive, this indicates we have movement away from the equilibrium point along at least one of the eigenvectors. The other vector has no movement (0 as the eigenvalue), but this one positive value will ensure we have an unstable equilibrium. Again, think of the ball, the positive eigenvalue indicates the ground slopes downwards from the equilibrium point so a ball balanced on top of this hill will be very unstable.

Now let's do the second equilibrium. The one where $S = P$ and $H = 0$ (everyone is sick). Let's evaluate the Jacobian at this equilibrium:

$$\begin{bmatrix} -\alpha \times P & 0 \\ \alpha \times P & 0 \end{bmatrix}$$

Now let's find the eigenvalues for this matrix. Once we go through the math we get two eigenvalues: this time 0 and $-\alpha \times P$. Again, the 0 eigenvalue can be ignored as it does not cause growth or change. The second eigenvalue however is negative, indicating the system moves toward the equilibrium point again. Look back at our exponential growth phase planes. Negative coefficients indicate trajectories towards the equilibrium (create a cup for the ball). Thus this second equilibrium is a stable one.

It's time to look at a more complex example, we'll consider our predator prey model. First we calculate the Jacobian matrix for this model:

$$\text{Jacobian} = \begin{bmatrix} \frac{\partial}{\partial M}\alpha \times M - \beta \times M \times W & \frac{\partial}{\partial W}\alpha \times M - \beta \times M \times W \\ \frac{\partial}{\partial M}\gamma \times M \times W - \delta \times W & \frac{\partial}{\partial W}\gamma \times M \times W - \delta \times W \end{bmatrix} = \begin{bmatrix} \alpha - \beta \times W & -\beta \times M \\ \gamma \times W & \gamma \times M - \delta \end{bmatrix}$$

Now that we have the Jacobian, we'll evaluate it at the trivial equilibrium of $M = 0$ and $W = 0$. The resulting matrix is:

$$\begin{bmatrix} \alpha & 0 \\ 0 & -\delta \end{bmatrix}$$

The eigenvalues of this matrix are α and $-\delta$. Thus one of the eigenvectors approach the equilibrium and the other moves away from it. This means we have an unstable equilibrium, which is actually good news as it indicates that the two animal populations will not spontaneously go extinct.

Let's now evaluate the more complex equilibrium point we identified earlier of $M = \delta/\gamma$ and $W = \alpha/\beta$. First we calculate the Jacobian at this point:

$$\begin{bmatrix} 0 & \frac{-\beta \times \delta}{\gamma} \\ \frac{\gamma \times \alpha}{\beta} & 0 \end{bmatrix}$$

When we calculate the eigenvalues for this point we obtain $i\sqrt{\alpha \times \delta}$ and $-i\sqrt{\alpha \times \delta}$. Here the i indicates the imaginary number $\sqrt{-1}$. That's a little strange, so how do we interpret this? Well, it turns out that imaginary numbers in the eigenvalues indicate oscillations in the phase planes, thus this results means we have oscillations around the point of equilibrium. Since we have no real component in the eigenvalues, there is neither attraction towards the point of equilibrium or repulsion away from it so we have a stable oscillation around the equilibrium.

Of course we already knew that from our simulations, but this stability analysis allows us to mathematically determine this relationship, a capability that is a very powerful tool. The following table summarizes the different types of eigenvalues that can be found for a system with two state variables and their associated stabilities.

Real Parts	Imaginary Part?	Stability
Both Equal to 0	No	Neutrally Stable
Both Equal to 0	Yes	Stable Oscillations
Both greater than or equal to 0	No	Unstable
Both greater than or equal to 0	Yes	Unstable Oscillations
Both less than or equal to 0	No	Stable
Both less than or equal to 0	Yes	Damped Oscillations (Stable)

Analytical vs. Numerical Analysis

The majority of this book has been focused on the numerical analysis of models and the qualitative conclusions that can be drawn from these results. This chapter has introduced a set of analytical tools that can be used – for the most part – to analyze the same models we have presented elsewhere in the book. Now take a moment to reflect on these different forms of analysis and what each one can offer.

The great benefit of the analytical techniques we present here is that they can provide precise answers to the general behavior of the system. Most of these same answers can also be determined numerically (e.g. running the simulation many times and exploring the results), but those answers will be less precise and definite. If you manually attempt to explore the parameter space of your model, it is possible that you could miss some set of parameter values that will give you unexpected behavior. An analytical analysis may be fully comprehensive and can guarantee the completeness of your conclusions.

A weakness of analytical methods is that your model must be solvable analytically. This means that you will probably need to keep your model from growing too complex in order to keep it analytically tractable. Also, some common functions such as `IfThenElse` logic can make analytical work much more difficult. Further, some models may simply be impossible to analyze analytically and these insolvable models may in fact be very simple in practice. For example, any model containing both X and $\log(X)$ in the same equation will be intractable to many forms of analysis.

We think both analytical and numerical work has a lot of applicability in practice. We do worry, though, about some of the analytical models and work we see presented or published. Sometimes these models seem to us to be much too simple to adequately represent the system they are supposed to be modeling. True, analytically the results of the models appear elegant and clear, but if the model is too simple to be relevant these results have little use and may actually be very misleading in practice. We worry sometimes that a focus on analytical work[8] leads to modelers prioritizing analytical tractability over model utility in their decisions. We believe a focus on analytical results can lead to reductionist models with reduced practical utility and we caution modelers against becoming too focused on elegant solutions and the expense of relevance. Where available, more realistic models are preferable, even if they require numerical solutions than overly simplistic analytically solvable ones.

[8] And, rightly or wrongly, analytical work is generally considered more prestigious and "serious" than numerical work.

Chapter 11

Going Global

Engaging people to cause positive action and change is one of the key goals of systems thinking and modeling. The growth of the Internet has created amazing opportunities to reach out and connect with people in ways that have never before been possible.

The Internet makes it easy to share models with other people. Not only can you email a specific person the tables and graphs of a model's results, but you can also build webpages publishing these results to share with the world. What is more, these results do not have to be limited to static data. Using Insight Maker, you can include an interactive version of your model allowing others to experiment with it directly on your webpage. This can be done on any page you have rights to edit including your personal website, a blog, and a company's information page.

Furthermore, the information flow doesn't have to be one-way from you to others. In a webpage you can include a feedback or comment form that allows anyone to comment and share his or her thoughts on the model right next to the model itself. These comments can be saved directly on the page allowing other people to read them and enabling a discussion to form around the model. This creates many avenues for collaboration and learning that would simply be impossible without the Internet.

In this chapter, we will show you how to develop webpages to showcase your insights and models to the world. We'll also show how to include tools to engage viewers and start a dialogue about your models. Before jumping into the models themselves, we will lay the groundwork by introducing the basic principles of web development. Once we have introduced these key principles, we'll walk through two examples of developing interactive models.

The Web in a Nutshell

The World Wide Web is based on a collection of many different technologies that work together. When developing a webpage there are three major technologies

that you need to be familiar with: HTML, CSS and JavaScript. Each of these technologies or languages plays a different role in the development of a webpage.

Technology	Commonly Called	Usage
Hypertext Markup Language	HTML	Webpage Structure
Cascading Style Sheets	CSS	Webpage Style
ECMAScript	JavaScript	Webpage Interactivity

The web is interesting in that each of these technologies is based on old-fashioned, simple text files. You write HTML text files, you write CSS files, and you write JavaScript files[1]. You do not need any fancy tools to create these files. Any simple text editor will do. A web browser takes the simple instructions and code in these files and converts them to the rich interactive webpages you see when you browse the Internet.

When teaching web development many books and sources will recommend that you use some kind of interactive web site builder (like Adobe Dreamweaver http://www.adobe.com/products/dreamweaver.html). That is certainly a great way to get up and running, but ultimately you will find the approach very limiting. To truly harness the different tools offered to you by the Internet, you will need to have some understanding of the underlying technologies and be able to work with them directly. So, rather than using a website builder as a crutch, we recommend jumping right into learning HTML, CSS and JavaScript.

In the following sections we'll give you a brief introductions to each of these fundamental web technologies. This introduction will be rapid so please do not worry if you do not fully understand everything. But the please do your best to engage with this material as it will provide you with everything you need to know in order to be able to get the maximum out of our later examples of interactive modeling webpages.

HTML Basics

HTML defines the structure of a webpage or document. An HTML document is made up out of a set of *tags*. Each tag is enclosed in triangular brackets. For instance, there is a tag called "<hr>" that will create a horizontal division line in your document ("hr" is an abbreviation of "horizontal rule").

Many types of tags will consist of an opening and closing tag paired together. A closing tag is written the same as an opening tag except there is a also backslash

[1]Please note that when you write CSS and JavaScript, your text is case-sensitive. This means that "ABC", "abc", and "Abc" will all be understood differently. HTML, on the other hand, is case-insensitive. In HTML, "ABC", "abc", and "Abc" will all be understood to mean the same thing.

immediately after the first triangular bracket. For instance, you could use a pair of "..." tags to make some text bold:

```
This is some text. <b>This text is bold.</b> This text is not bold.
```

Some tags may also have "attributes" which modify the behavior of the tag. Attributes are included within the opening brackets of the tag after the tag name. For instance, the "<a>" tag is used to make links between webpages. The "" tag has an attribute "href"[2] which is the URL the link should connect to. The following HTML creates a link to Google:

```
If you ever need to search something, just go
to <a href="http://Google.com">Google.</a>.
```

Every HTML page contains some general boilerplate that structures the document. This boilerplate will look almost identical from webpage to webpage and it contains several unique tags which split the document into two sections: a "head" section to store the page title and page keywords for search engines, and a "body" section which contains the page content that is what is actually shown to the user. You will spend most of your time editing the body section. The standard template for a webpage is as follows:

```
<html>
<head>
    <title>A Sample webpage</title>
</head>
<body>
    Document contents goes here...
</body>
</html>
```

There are dozens of different tags you can use in your document to structure it. We can't cover them all here, but the following table summarizes a few of the most useful ones:

Tag	Usage	Example

[2]The tag name "a" comes from "anchor" and "href" is an abbreviation of "hyperlink reference". Many of the conventions with web development may seem strange and so you should understand the long history of these technologies and the resulting historical baggage that comes with them.

a	Creates a link	Google.
b	Makes text bold	This text is bold.
i	Makes text italic	This text is italic.
u	Makes text underlined	This text is <u>underlined</u>.
center	Centers a paragraph	In the middle.
p	Creates a paragraph of text	This is a paragraph.
hr	Creates a dividing line	Something <hr> Something Else
h1	Creates a heading	This is a Heading
img	Embeds an image	<img src="http://example.com/image.png">

We can combine these tags together to form more complex documents. The following is an example of a full-featured webpage.

```
<html>
<head>
    <title>A Sample webpage</title>
</head>
<body>
    <h1>Introduction</h1>
        <p>Here is some information about my page.</p>
    <h1>The Content</h1>
        <p>Here we have the meat of the page.</p>
    <hr>
    <h2>For Further Information</h2>
        <p>Here we have links to other sites about this content:<p>
        <p>We could check out <a href="http://BeyondConnectingTheDots.com">
            this book's site</a> for instance.</p>
</body>
</html>
```

Open whatever word processor you use on your computer and save this to *MyPage.html* as a plain text file[3]. You can then open this file in your web

[3]Webpages are always stored as plain text. This differs from, for instance a Microsoft Word document (".doc" or ".docx" extension) or a Rich Text Format document (.rtf extension). You need to ensure you save your document as a plain text document with the extension ".html" or ".htm". You can use any text editor you want, but if you get an editor designed for writing webpages it will have helpful features such as coloring your tags differently from the standard text as you edit the webpage. We recommend Sublime Text (http://www.sublimetext.com/) as a high quality editor for serious work.

browser (Internet Explore, Firefox, Chrome, Safari, etc.). Experiment by adding some more paragraphs and formatting to see how the document changes.

For more information and tutorials on HTML, we recommend the Mozilla Developer Network's guides (https://developer.mozilla.org/en-US/docs/Web/HTML).

CSS Basics

Where HTML is used to define the structure of a document, CSS is responsible for styling this structure. This styling includes aspects like font and color choices in addition to the general layout. A CSS document is a list of rules where each rule has two parts: a selector that tells the browser what elements of the page the rule applies to, and a set of styles that tells browser how to style those elements. For example, take the following CSS code.

```
p {
    margin: 20px;
}

h1, h2 {
    font-size: 72px;
    color: red;
}
```

This code has two rules. In the first rule the selector is "p" meaning the rule will apply to all "<p>" tags in the document. The styling for this rule says to apply a 20-pixel margin around each of these paragraph tags. The second rule has the selector "h1, h2". This means apply the rule to both "<h1>" and "<h2>" tags and to set the contents of those tags to have an extra large font and to be colored red.

There are numerous different aspects of an element's style you can set with CSS. For a full and detailed reference we recommend the Mozilla Developer Network's coverage of CSS (https://developer.mozilla.org/en-US/docs/Web/CSS/Reference).

CSS for a webpage can be placed in a standalone file which is referenced by the webpage or it can be included directly within the webpage. Both these can be accomplished by placing a CSS rules within a special tag in the *head* section of the document. For example, taking the *head* section from our earlier document, we could either embed the CSS directly:

```
<head>
    <title>A Sample webpage</title>
    <style>
```

```
        p {
            margin: 20px;
        }
        h1, h2 {
            font-size: 72px;
            color: red;
        }
    </style>
</head>
```

Alternatively we could save the CSS to an external text file (such as *MyStyles.css*) and link to it in the *head* of our document:

```
<head>
    <title>A Sample webpage</title>
    <link rel="stylesheet" type="text/css" href="MyStyles.css">
</head>
```

JavaScript Basics

JavaScript[4] provides interactivity for webpages. JavaScript is a powerful programming language that you can use to respond to user actions, run calculations, or modify a webpage. An example of using JavaScript code to calculate a Fibonacci number[5] is below:

```
function fib(n){
    if(n==1 || n==0){
        return 1;
    }
    return fib(n-1) + fib(n-2);
}

alert("The tenth Fibonacci number is: "+fib(10));
```

Like CSS, there are two ways to embed JavaScript into an HTML document. The first is to include the JavaScript directly in the document like we did for the CSS:

[4]The name "JavaScript" is a source of perpetual confusion. What we know colloquially as JavaScript is officially called ECMAScript. Due to trademark issues Microsoft refers to it as JScript when you are using Internet Explorer. It is important to note that *JavaScript* and *Java* are different technologies. They share part of a name due to historic branding purposes but they are completely different languages.

[5]Where the first two Fibonacci numbers are 1 and the Fibonacci numbers thereafter are the sum of the two preceding numbers. The Fibonacci sequence begins: 1, 1, 2, 3, 5, 8, 13, 21, 44...

```
<head>
    <title>A Sample webpage</title>
    <script>
        function fib(n){
            if(n==1 || n==0){
                return 1;
            }
            return fib(n-1) + fib(n-2);
        }

        alert("The tenth Fibonacci number is: "+fib(10));
    </script>
</head>
```

The second method to include the code is to save the JavaScript into a text file (such as *MyScript.js*) and link to it in the document:

```
<head>
    <title>A Sample webpage</title>
    <script src="MyScript.js"></script>
</head>
```

JavaScript is a very powerful tool but also a very complex one. This chapter will illustrate usages of JavaScript but we cannot hope to teach you how to write new JavaScript yourself in this single chapter. Again, we refer you to the Mozilla Developer Network to learn more about JavaScript (https://developer.mozilla.org/en-US/docs/Web/JavaScript).

Creating an Webpage for Engagement

Now that we have made it through some of the technical details, let's jump into building a webpage for an interactive model that users can comment on. There are three basic things we want this webpage to have:

1. A description of the challenge we are tackling, why we built the model, and what the model contains.
2. An interactive version of the model that the user can explore and run simulations with.
3. A discussion forum about the model where users can post comments and see what others have posted.

This might seem ambitious, and it is! But using freely available technologies and services we will be able to put this webpage together very quickly. Let us split the process of developing the webpage into three steps: first we'll create

the general page framework, then we will add the interactive model, and lastly we will add the discussion forum.

Creating the Page and Description

Assume we decide we want to create a webpage exploring population growth and whether the Earth can sustain humanity into the future. We start building our webpage by creating an HTML file and putting the following text in it.

```
<html>
<head>
    <title>A Fragile Future</title>
</head>
<body>
    <h1>Introduction</h1>
        <p>This is a model of world population
            changes into the future.</p>

    <h1>The Model</h1>
        [Model goes here]

    <h1>Discussion</h1>
        [Discussion forum goes here]
</body>
</html>
```

This creates a page with three sections: Introduction, The Model, and Discussion. We can fill in the Introduction section with text describing the problem we face and our approach to understanding it in our model. In this example page, we have just written a single sentence but you could extend it with more details on the model to fully explain to the viewer why this is important and how we have modeled it.

The placeholders [Model goes here] and [Discussion forum goes here] are where we will insert our model and discussion forum later on. For now though, we just want to layout the structure of the page.

Adding an Interactive Model

Now that we have created the structure for our webpage, we can add the interactive model. There are several ways to do this. One way would be to write the model in JavaScript and include it directly in the webpage. JavaScript is a full-featured programming language and could be used to implement any of the models described in this book. Although implementing a model in JavaScript is definitely possible, it would require a lot of work. Writing a model in JavaScript

would take a large amount of time and would not be possible without extensive programming experience.

Fortunately, using Insight Maker there is a much easier approach. Insight Maker models can be easily embedded in a webpage without any special effort on your part. So rather than writing our world population model in JavaScript, we can simply build the model in Insight Maker and then embed the resulting model in our webpage. So build your model in Insight Maker just as you would build one normally. You can also take an existing model that you have already built and use that one. For this example, we will use the World3 model (http://InsightMaker.com/insight/1954) which has a detailed worldwide model of population change[6].

Once you have finished constructing your model, click the **Embed** button in the **Tools** section of the Insight Maker toolbar. A window will open containing HTML code that you should paste into your webpage. This code will embed a version of the insight when it is placed in a webpage document. For the World3 model this code is something like:

```
<IFRAME SRC="http://InsightMaker.com/insight/1954/embed?topBar=1&sideBar=1&zoom=1"
TITLE="Embedded Insight" width=600 height=420></IFRAME>
```

Take this code and use it to replace the [Model goes here] placeholder in your webpage. Save the webpage and open it in a browser and you will now have a rich interactive version of your model embedded directly in your webpage!

There are several features of the embedding that you can control by editing the "<IFRAME>" tag. For instance the "width" and "height" attributes control the size of the embedded model. They are specified in pixels and you may change them to make the embedded model smaller or larger. The "topBar" and "sideBar" parts of the URL control whether the toolbar and the sidebar will be shown in the embedded model's interface. By default, they are set to 1 indicating these elements will be shown. Set them to 0 to hide the bars when the model is displayed. The "zoom" part determines whether the model diagram is shown at its full size or if it is zoomed to fit the window (the default). Set this to 0 to prevent the model diagram from automatically being resized to fit the window.

Adding a Discussion Section

Now we have one last piece to add before we have completed our webpage. We want people to be able to carry on a discussion about the model directly within the page. To make this possible, we need to add some sort of forum or discussion software.

[6]This model was described and discussed in detail in the book *The Limits to Growth*

We could program our own custom discussion system; but, like the case of the model itself, this is a place where it is easier to leverage existing free software than it is to develop our own. A number of free commenting and discussion systems are available. One of these is called Disqus (http://disqus.com). If you read a number of different news sites or blogs you have probably already used Disqus as many sites utilize their software.

You will need to sign up for a Disqus account to be able to embed their discussion software, but fortunately like Insight Maker it should not cost you a thing. Once you have completed signing up at http://disqus.com, follow the site for directions on how to embed Disqus in your own webpage. You should be given code that looks similar to the following to place into your webpage:

```
<div id="disqus_thread">Discussion Here</div>
<script type="text/javascript">
    var disqus_shortname = 'SHORT-NAME-DEMO'; // required: replace example with your f
    (function() {
        var dsq = document.createElement('script'); dsq.type = 'text/javascript'; dsq.
        dsq.src = '//' + disqus_shortname + '.disqus.com/embed.js';
        (document.getElementsByTagName('head')[0] || document.getElementsByTagName('bo
    })();
</script>
```

First edit this code as instructed (e.g. replace any usernames or ids with the ones you have been provided by Disqus) and then replace the [Discussion forum goes here] placeholder in your page with this code. Load the page and test to see if it is working. One issue with Disqus is it might not work if the webpage is being opened from a file on your computer. You may need to upload it to the domain name you entered when you signed up for Disqus to ensure it works correctly.

Completed Page

We have just put together a powerful site very quickly. Our site lets us share an interactive model with people anywhere in the world and allows them to comment directly on the model. All that it took to do this was the following completed code:

```
<html>
<head>
    <title>A Fragile Future</title>
</head>
<body>
    <h1>Introduction</h1>
        <p>This is a model of world population
```

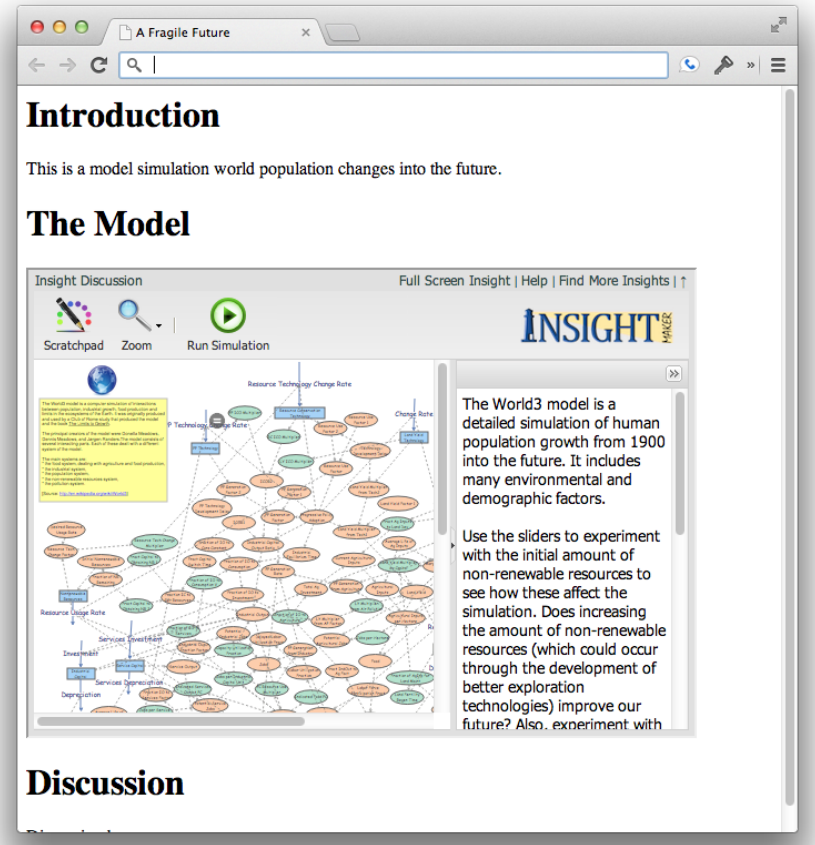


Figure 1. Completed page with embedded model.

```
        changes into the future.</p>

<h1>The Model</h1>
    <IFRAME SRC="http://InsightMaker.com/insight/1954/embed?topBar=1&sideBar=1&zoom=1"
    TITLE="Embedded Insight" width=600 height=420></IFRAME>

<h1>Discussion</h1>
    <div id="disqus_thread">Discussion here</div>
    <script type="text/javascript">
        var disqus_shortname = ''; // required: replace example with your forum shortname
        (function() {
            var dsq = document.createElement('script'); dsq.type = 'text/javascript'; dsq.as
```

```
                dsq.src = '//' + disqus_shortname + '.disqus.com/embed.js';
                (document.getElementsByTagName('head')[0] || document.getElementsByTag
            })();
        </script>
</body>
</html>
```

A working version of this site may be viewed at http://BeyondConnectingTheDots.com/book/embedded-model/. There is a lot more we could do with the site. Spend some time now experimenting with it. Add some more descriptive text, maybe add some images, and try to use CSS to adjust the styling.

Flight Simulators and Serious Games

In the preceding section we described how to rapidly develop a website that contains an interactive model and provides users the ability to comment and discuss the model directly on the page. By leveraging Insight Maker, we were able to get an interactive version of our model embedded in our webpage just by copying a few lines of code. By leveraging Disqus, we were able to include a discussion forum with a similar amount of effort.

In many cases, what we created may be exactly what you are looking for. However, in other cases it is possible that you will wish to provide your users with a unique experience tailored to understanding a specific problem. For instance maybe you would like to develop what is known as a "flight simulator", a simulation tool that puts the user in the position of trying to manage a problem or achieve an outcome. For example, if you had a model of a business going through a disruptive change, you could place the user in the position of the company's leader and with instructions to adjust parameters in the model in order to safely shepherd the company through this challenge.

Similarly, "serious games" are tools designed to both engage and educate about a system. You can create a simulation model at the heart of a serious game or a flight simulator. You may give users direct access to this simulation model's interface, but generally you will want to build a custom interface on top of the model that hides the stock and flow diagram and instead displays a control panel type interface to the user.

Fortunately, web technologies provide a rich environment for developing these flight simulators and serious games. Furthermore, using Insight Maker you can build your model and simulation engine using its model building tools and then build a custom interface on top of the model to provide the exact experience you want to the user. In the following sections we will develop a custom interface to control our world population simulation.

Setting up the Page

We'll start by stripping down our page from the previous example. Let's remove the commenting system and the introduction so the page just contains the model (you can add these other items back later on your own as an exercise). After we do this, we will be left with a page just containing the embedded world simulation model.

In this case, however, we do not want the user to actually interact with or even see the embedded model. We will be adding our own custom interface and just using the embedded model to run simulation in the background. To hide the embedded model we can add a CSS rule that makes the <iframe> tag invisible:

```
iframe {
    display: none;
}
```

This rule turns off the display of all <iframe> tags in the page. They are still there and in the page, but they are not shown to the user. The resulting completed template for our page is printed below. When you open this in your browser you should see a completely blank webpage.

```
<html>
<head>
    <title>A Fragile Future</title>
    <style>
        iframe {
            display: none;
        }
    </style>
</head>
<body>
    <IFRAME SRC="http://InsightMaker.com/insight/1954/embed?topBar=1&sideBar=1&zoom=1"
    TITLE="Embedded Insight" width=600 height=420></IFRAME>
</body>
</html>
```

Creating the Control Panel

HTML has a tag called "" that lets you create form elements for users to input data. The <input> tag has an attribute called "type" that determines what the type of the input element will be. There are a wide number of types including "number", "text", "color", "textarea", "date", and "button". For our control panel, we'll design it to modify two parameters of the model and to have a button users can press to run the simulation. In addition to specifying the type

of the inputs, we should also specify their initial values in the control panel. We can do that using the "value" attribute of the <input> tag.

Finally, we will need some method to reference the inputs and to load their values later on. Each tag in an HTML document has an optional "id" attribute. This attribute can be used to obtain a reference to that element from JavaScript. We'll set the id attribute for our two input fields so we can obtain their values when we are ready to run the simulation.

The resulting control panel will look something like the following code. As you can see we have presented the user with a simple task, to find a combination of settings that results in over 5 billion people in the year 2100 (which is in fact a significant decrease from the current population size so it should not be too hard). You should place this code after the <iframe> tag in your document.

```
<center>
    <p>This is a game to keep the world's population larger than 5 billion in the year
        We can experiment with the amount of non-renewable resources in the world and
        start year for a clean energy eco-friendly policy.</p>
    <p> Initial Non-Renewable Resources: <input type="number" value="100" id="resource
    <p> Start Policy Year: <input type="number" value="2013" id="year" /> </p>
    <p> <input type="button" value="Test Scenario" /> </p>
</center>
```

This will create two input fields allowing users to input numeric values. The first, *Initial Non-Renewable Resources* will allow the user to increase or decrease the amount of non-renewable resources assumed in the model at the start of the simulation. The second, *Start Policy Year* allows the user to specify the start date to implement a clean technology policy which will reduce the amount of pollutants being generated in the simulation. A button is also created that lets the user test the scenario in the simulation.

Making it Interactive

We use JavaScript to add interactivity to the webpage. Let's define a JavaScript function *testScenario* that we will use to first read in the user specified options from the control panel, then run the simulation with these parameter values, and finally report to the user whether or not they were successful in keeping the population size above 5 billion.

We will fill out the *testScenario* function with steps later; but for now, just add the following code to the head section of your webpage.

```
<script>
    function testScenario(){
        alert("Scenario tested!");
```

```
    }
</script>
```

This creates the function, but we also need a way for the function to be executed when the "Test Scenario" button is pressed. There are several ways to do this. The easiest is to set the "onClick" attribute of the button to call the function. The "onClick" attribute of an input may contain JavaScript code that is executed when the button is clicked. To link up our button with the *testScenario* function, we change our input button in the HTML to:

```
<p> <input type="button" value="Test Scenario" onclick="testScenario()" /> </p>
```

Implement the webpage up to this point and check to make sure that you see a message pop up saying "Scenario tested!" when you press the "Test Scenario" button.

Now that we have implemented basic interactivity, let's flesh out the *testScenario* function.

Load Parameter Values from the Control Panel

To access an input field from JavaScript we use the *document.getElementById* function. This function is built into your browser and allows you to obtain a reference to one of the input elements based on its "id" attribute. Once we have a reference to the input element we can use the element's "value" property to obtain the number the user has entered into the input field.

The following code defines two variables in JavaScript with the same values as the ones the user has entered. Enter this code at the top of your *testScenario* function.

```
var resources = document.getElementById("resources").value;
var year = document.getElementById("year").value;
```

Inject the Parameter Values into the Model

Insight Maker has an extensive JavaScript API[7] that can be used to modify and script models. This is the same API that may be used with Button primitives. Refer to the API reference at http://insightmaker.com/sites/default/files/API/files/API-js.html for full details about the API.

The API instructions provide examples about how to integrate and modify an embedded model. We will adapt those instructions to our own case. First, as the instructions indicate we need to update our <iframe> tag to add an "id" attribute. We adjust our <iframe> tag like so:

[7] An API, or Application Programming Interface, is a set of commands and functions that can be used to interface programmatically with an application.

```
<IFRAME id="model" SRC="http://InsightMaker.com/insight/1954/embed?topBar=1&sideBar=1&
TITLE="Embedded Insight" width=600 height=420></IFRAME>
```

Now we can obtain a reference to the model using the *document.getElementById* function from before and then we can send API commands to it using its *postMessage* function. Within Insight Maker, we use the *findName* API command to get a reference to a specific primitive and then use the *setValue* API command to set the value of that primitive to the value of the parameter in the control panel. Add the following code to the *testScenario* function.

```
var model = document.getElementById("model").contentWindow;

model.postMessage("setValue(findName('Initial Nonrenewable Resources'), '"+(resources/
model.postMessage("setValue(findName('Progressive Policy Adoption'), '"+year+"')", "*"
```

This convoluted *postMessage* mechanism to pass JavaScript commands to the embedded model is a constraint necessitated by your browser's security mechanisms. It makes the processing of interacting with embedded models more complex than we would like, but fortunately it is still possible to do everything that we need to do even using it.

Run Simulation and Access Results

To run the model, we use the *runModel* Insight Maker API command. We indicate that the simulation should be run in "silent" mode so the results are returned[8]. We then use the *lastValue* function to obtain the final population size for the simulation in the year 2100. Copy this into your webpage at the end of the *testScenario* function:

```
model.postMessage("runModel({silent: true}).lastValue(findName('Population'))", "*");
```

So far we have just demonstrated one-way communication between the control panel and the embedded model. This is the first point in time when we need to be able to communicate the other way: to receive data back from the embedded model.

Unfortunately, due to the security constraints imposed by your browser, this is slightly complex. In order to receive a message back from the embedded model, we need to register an event handler with your main browser window. Don't worry if you don't fully understand this, just copy the code below into the script tag of your window.

[8]There are two primary ways of running Insight Maker models using the *runModel* API command. One is the regular way where a results diagram will be shown but the results will not automatically be returned in JavaScript. The second way is in silent mode where the results are returned, but results graphs are not shown in the model interface.

```
function scenarioComplete(event)
{
    if(event.data){
        var pop = Math.round(event.data);
        if(pop > 5000000000){
            alert("You won! The population size of "+pop+" is larger than 5 Billion!");
        }else{
            alert("You failed! The population size of "+pop+" is smaller than 5 Billion!");
            alert("Please try again.");
        }
    }
}

window.addEventListener("message", scenarioComplete, false);
```

Final Result

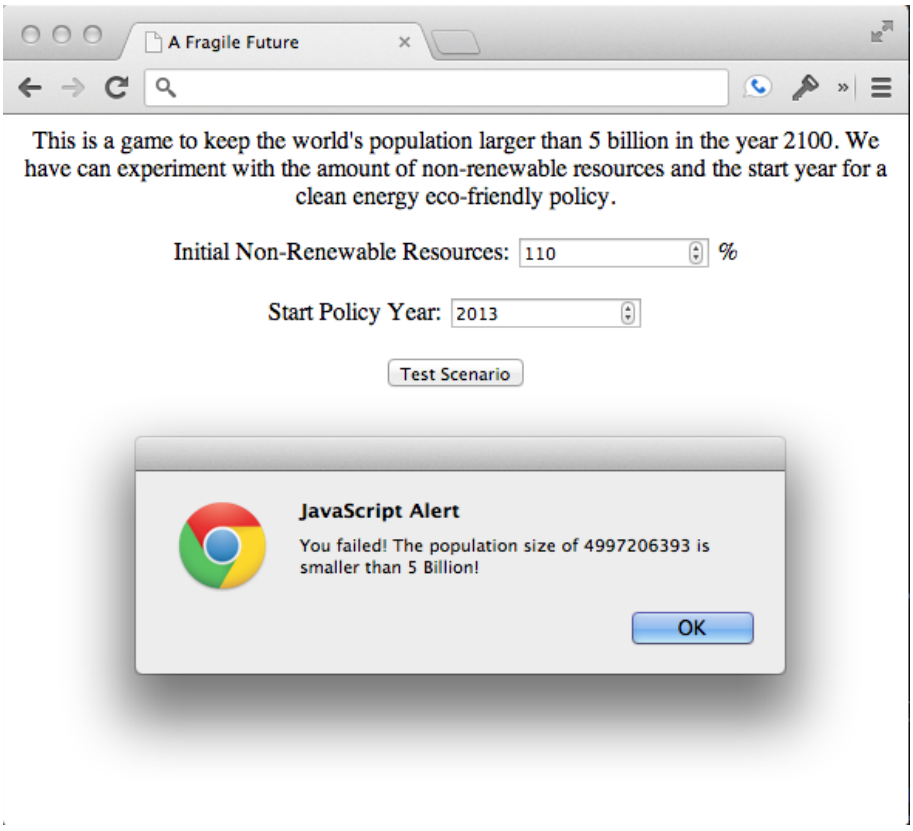


Figure 2. Completed control panel.

The code for the completed webpage is provided below and a working version of the page may viewed at http://BeyondConnectingTheDots.com/book/control-panel/.

```
<html>
<head>
    <title>A Fragile Future</title>
    <style>
        iframe {
            display: none;
        }
    </style>
    <script>
    function testScenario(){
        var resources = document.getElementById("resources").value;
        var year = document.getElementById("year").value;

        var model = document.getElementById("model").contentWindow;

        model.postMessage("setValue(findName('Initial Nonrenewable Resources'), '"+(re
        model.postMessage("setValue(findName('Progressive Policy Adoption'), '"+year+"

        model.postMessage("runModel({silent: true}).lastValue(findName('Population'))"
    }

    function scenarioComplete(event)
    {
        if(event.data){
            var pop = Math.round(event.data);
            if(pop > 5000000000){
                alert("You won! The population size of "+pop+" is larger than 5 Billio
            }else{
                alert("You failed! The population size of "+pop+" is smaller than 5 Bi
                alert("Please try again.");
            }
        }
    }

    window.addEventListener("message", scenarioComplete, false);
    </script>
</head>
<body>
    <IFRAME id="model" SRC="http://InsightMaker.com/insight/1954?embed=1&topBar=1&side
    TITLE="Embedded Insight" width=600 height=420></IFRAME>

    <center>
```

```
        <p>This is a game to keep the world's population larger than 5 billion in the year 2100.
            We can experiment with the amount of non-renewable resources in the world and the
            start year for a clean energy eco-friendly policy.</p>
        <p> Initial Non-Renewable Resources: <input type="number" value="100" id="resources" /> %
        <p> Start Policy Year: <input type="number" value="2013" id="year" /> </p>
        <p> <input type="button" value="Test Scenario" onclick="testScenario()" /> </p>
    </center>

</body>
</html>
```

The key goal of this chapter is not that you completely understand this, but rather that you will be able to adapt it to your own needs. There are a lot of additional changes that could be made to this demonstration. You could clean up the control panel and make it look more attractive by adding some CSS rules. You could add additional inputs to control other parts of the model. You could show the user the trajectory of the population instead of just the final value. Go ahead and experiment with this example to see what you can make it do.

Additional Tips

Web development is a very complex topic with a lot of nuances. The preceding sections should have given you a brief introduction in how to create interactive models for engaging an audience and encouraging discussion and learning. Although we cannot give you a comprehensive course in web development, there a few additional web development tips that will be very useful when you start to develop your own webpages.

Frameworks and Toolkits

Making an attractive web application is hard. Admittedly, the control panel application we developed does not look very good. We could spend some time improving its appearance by adding additional CSS rules but since we are not professional designers it is quite possible that the results of our efforts would only look amateurish and unattractive. Additionally, writing JavaScript to interact with webpages is also hard. These web technologies were developed over decades and many of the functions and techniques that need to be used are slightly archaic and are difficult to learn.

Fortunately, a number of toolkits and frameworks have been developed that make it easier to develop powerful and attractive web pages and control panels. Below we highlight some important toolkits that you might want to explore and considering adopting for your own usage. These toolkits can be embedded within your webpage extending its functionality. They will help you make

more attractive and powerful applications quicker. The ones listed are all also available under open source licenses allowing you to use them for free.

Twitter Bootstrap (http://GetBootstrap.com) : Bootstrap is a framework for developing attractive webpages. It has many tools and rules that can be combined together to create visualizing pleasing webpages with minimal effort. If you don't have a good sense of design, Twitter Bootstrap could be a great help to you.

JQuery (http://jquery.com/) : JQuery is a library designed to improve the JavaScript functions needed to interact with a webpage. It generally greatly simplifies and reduces the number of keystrokes you need to carry out some task. For instance "document.getElementById('item')" becomes in jQuery simply "$('#item')".

JQuery UI (http://jqueryui.com/) : A spin-off from the JQuery project, this toolkit provides control panel elements that are themable and more extensive than the built-in <input> tags. Grids, sliders, and more are all available from this project.

ExtJS (http://www.sencha.com/products/extjs) : ExtJS is a comprehensive library for developing powerful applications. It has extensive tools to develop interface and control panels. It is also what is used to develop Insight Maker's interface.

Debugging Webpages

As you develop your webpage, it is almost certain that you will make many mistakes and typos as you go along. If you make a mistake within the HTML or CSS of the page you will have an immediate visual indication that something is wrong and you can experiment with your code until it is fixed.

JavaScript errors, on the other hand, generally won't provide any visual feedback that an error has occurred. The most likely indication that a JavaScript error has occurred is that nothing happens when you click a button or expect an action to occur. Debugging issues like this can be quite difficult. Fortunately, with just a little bit of additional work you can get access to very rich and informative JavaScript error messages letting you know exactly what went wrong and when.

There are two approaches to obtain access to the JavaScript error messages. The first is to actually edit your webpage and add code to show an error message when an error occurs. Adding the following to a <script> tag in the head section of your document will do that:

```
window.onerror = function(message, url, line) {
  alert("JavaScript Error: \n\n" + message + " (line: " + line + ", url: " + url + ")"
}
```

Now when an error occurs, an alert will pop up with a brief description of the error and information about where it occurred in your code.

The second approach you can take is to use the developer tools that are built into your web browser to study the webpage and observe errors as they occur. Excellent developer tools are built into all modern browsers. These tools let you study the structure of the webpage, profile the performance of your code, and examine how the webpage behaves.

One particular tool is very useful when developing webpages: the JavaScript console. Once you have opened the JavaScript console (search on-line for the exact directions on how to do this for your specific web browser) errors and messages from the webpage will appear in the console as they occur. What's more, the console allows you to evaluate JavaScript commands in the webpage simply by typing the commands into the console.

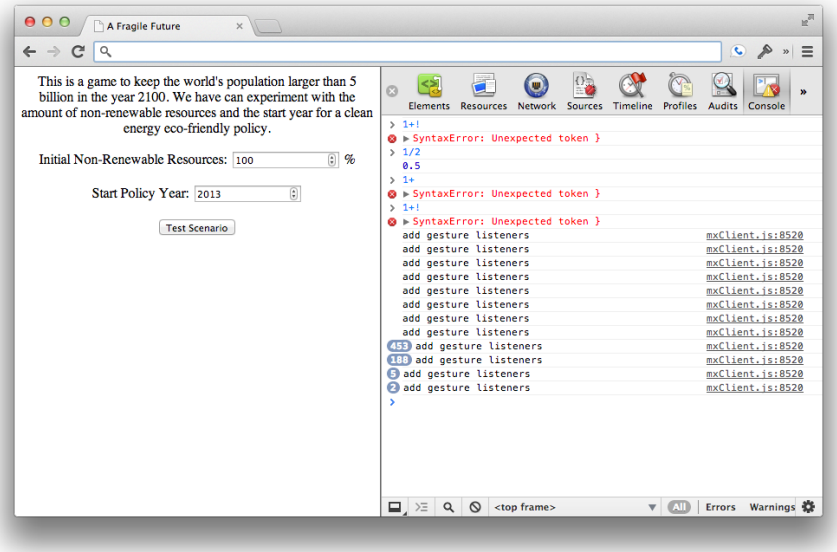


Figure 3. Google Chrome's JavaScript console.

One approach to debugging code is to put *alert* functions into the code updating you on the progression of the code or displaying values of the JavaScript variables. This works, but can be very clumsy and disruptive. When you have the console open, a better approach is available to you. You can send messages directly to the console providing information on the status of the program. For example:

```
console.log("The value of the variable is: " + myVariable);
console.error("An error has occurred!");
```

Sending Complex Data Back and Forth

The *postMessage* communication technique to send data back and forth to the embedded model can only be relied upon to send strings and objects that can easily be converted to strings (like numbers)[9]. Oftentimes you will want to pass more complex data from the simulation to the containing window. For instance you might want to pass the whole time series of values one or more primitives took on over the course of the simulation.

To handle these more complex objects you must convert them to strings. JavaScript provides a number of techniques to do so. For instance, if you have an array, you can convert it back and forth from a string using the *join* and *split* functions:

```
var data = [1, 4, 9, 16, 25];
var str = data.join("; "); // "1; 4; 9; 16; 25"
str.split(", "); // [1, 4, 9, 16, 25]
```

By far the most useful and flexible method of converting JavaScript objects to and from strings are the JSON commands. JSON, JavaScript Object Notation, is a general file format for storing data. It is based on the standard method for declaring JavaScript objects (e.g. {key: value}) but has some differences. What is great about JSON is that your browser already has built-in commands for converting JavaScript objects (a number, array, or other object) into a string and then later back into an object.

You can use this technique to send arbitrarily complex objects back and forth from your simulation to your webpage. Let's see how the JSON commands works:

```
var obj = {title: "I'm a complex object", data: [1, 4, 9]};
var str = JSON.stringify(obj); // '{"title":"I'm a complex object","data":[1,4,9]}'
JSON.parse(str); // {title: "I'm a complex object", data: [1, 4, 9]};
```

Hosting a webpage

In this chapter, we saved the webpages we have created to our personal computers' hard drives and opened them in a browser from there. This works great for development, but it does not allow us to share our creations with others.

Once you are ready to publish your webpages, you must move the HTML, CSS and JavaScript files off your computer and onto a web-server or web-host so that others can access them over the Internet. There are a number of options for web-hosting that range from the simple to the complex and from the free to the expensive.

[9] The specification for this feature provides that any type of JavaScript object should be supported, however a number of recent browsers only support strings.

On the simple and free end of the spectrum there are free blogging sites like Blogger (http://www.blogger.com) or WordPress (http://wordpress.com/). These sites allow you to create free blogs but they also allow you to do much more than that. These types of sites will generally let you edit the source HTML of your pages allowing you to implement the demos in this chapter directly within a blog post.

A step up from simple sites like these blogging platforms are shared hosting providers. Shared hosting providers such as DreamHost (http://dreamhost.com) take a server and allow multiple people to purchase space on the server to run their webpages. There are numerous shared hosting providers available. A more advanced version of shared hosting is Virtual Private Server (VPS) hosting. VPS providers such as RimuHosting (http://rimuhosting.com/) are similar to shared hosting providers in that they fit many customers on a single server. Where they differ is that a VPS host will give each customer a virtualized computer. Each individual customer will feel like they have complete control over their own computer and operating system even though they are sharing the actual hardware with others.

At the high end of the spectrum of complexity, cost and power are dedicated servers. In this case you purchase or rent a machine dedicated solely to the hosting your projects. This gives you complete control of your hosting situation but is expensive and may take a lot of effort to set up and maintain.

In general, we recommend starting small. Sign up for a Blogger account and experiment with these techniques there. If you keep at it and your site grows, at some point you will outgrow this simple solution and at that time you can upgrade to a more advanced hosting solution.

Chapter 12

Modeling with Agents

The modeling techniques we have taught up until this point have focused on gaining insights using highly aggregate models of a system. This means that when we looked at models of population growth, we did not explore individual people and instead focused on the population as a whole. This high-level aggregate approach to modeling helps us cut through unnecessary details to understand the core dynamics of a system and it is often quite effective.

For certain models however, this high-level view may hamstring our abilities to explore important questions. For instance in a disease model we may care about the physical relationship between people in the model. Are they near each other? How often do they come into contact? Can we attempt to control the disease by manipulating how they move about and relate each other? These are all questions that are very hard to answer with a standard System Dynamics model.

Heterogeneity, differences between individuals, is difficult to represent using System Dynamics models. One approach that is used is simply to duplicate the model structure for each different class of person or entity in the model. We recall seeing one model that explored education in the United States. The authors wanted to explore the differences between male and female students. To do so, they simply copy and pasted the entire model structure (consisting of dozens of stocks and flows) and configured one of these copies for male students and the other copy for female students.

Granted, this approach can be made work, but it requires a lot of effort to set up and configure even in the simple two gender case. When you have more than two cases it can quickly become completely unmanageable. Furthermore, duplicating parts of your model is a recipe for creating unmaintainable models afflicted by bugs. The reason for this is that when you later make changes to your model, you are going to need to ensure the changes are made correctly to everyone of the separate model copies. Although simple in principle, in practice this is very easy to mess up and is an direct route for bugs to be introduced into the model.

Fortunately, alternative modeling paradigm to System Dynamics exists that is excellent for modeling discrete individuals. It is called Agent Based Modeling and is focused on simulating individual agents and the interactions between these agents[1]. In this chapter will introduce Agent Based Modeling and show how you can use it to explore questions that cannot be answered with pure system dynamics.

The State Transition Diagram

Up until now, are primary modeling tool as been the stock and flow diagram. This type of diagram is very useful for summarizing large aggregate systems. The stock can model entities that can take on a range of values and flows are well suited for specifying the changes in these values.

In addition to representing aggregate systems, stock and flow diagrams can also be used to model things on an individual level. For instance, a model of a persons motivations could be represented using a stock and flow diagrams. The strength or importance of each type of motivation – money, family, etc. . . – could be represented as a stock with flows changing the importance of the motivations over time.

When looking the the individual scale however, we will oftentimes find ourselves wanting to defined characteristics of the individual using simple on/off logic. For instance, take this issue of an individuals sex. We this can be represented using two categories Male or Female (leaving aside transgendered individuals for the sake of simplicity). Similarly, when constructing a model of a disease, we might want to say a person is either sick or not sick (with no nuances such as slightly sick or highly sick). You could attempt to represent these different categories using stocks, but the formulation and equations to do so will be unnecessarily complicated.

Where the stock and flow diagram is used to model changing systems with continuous variables, the state transition diagram is used to model systems with discrete variables. Within Insight Maker, state transition diagrams are constructed almost the same way as stock and flow diagrams except all stocks are replaced with *State* primitives and all flows are replaced with *Transition* primitives. Both the state primitives can be added to the model by right-clicking on the model diagram. Transition primitives will automatically be created with you connect to state primitives together use the standard "Flow" connection type.

[1] System Dynamics also has a another standard tool for dealing with heterogeneity. This tool called "arrays" or "indexing" and allows you to transparently create multiple copies of your model during simulation to match different classes. Arrays are not as flexible as fully agent based models though. If you consider the continuum of fully aggregate System Dynamics models on one end to fully individualized Agent Based Models on the other, you can think of arrays as existing part way along this continuum.

A state primitive is possibly the most simple primitive available as it can only take on one of two values: true or false. When the state value is true, the state is active. When the state value is false, the state is not active and the agent does not occupy that state. When configuring a state primitive, you only need to set whether the state is initially active or not at the start of the simulation. This can simply be `true` or `false`, but it can also be a logical equation the depends or other primitives in the agent. For example if you had a variable in the agent called [**Size**], and you wanted a state to be initially active is the value of [**Size**] was greater than 5, you could put the following as the initially active property for the state: `[Size] > 5`.

A transition primitive moves an agent between states. For instance, if you had to states in your model *Healthy* and *Sick*, you could have one transition primitive moving agents from the healthy state to the sick state and another transition primitive moving them the other way.

There are three different ways a transition from one state to another can be triggered:

Timeout : In this mode the transition will be triggered a specific amount of time after that first state becomes active. For instance, if we had a disease model where the disease lasted 10 days, we could have the transition from the sick to healthy state have a time out of 10 days.

Probability : In this mode there is a probability of the transitioning happening each time period. For instance, in the disease model if the disease only lasted 10 days on average put could last longer or short you could use a transition probability of 0.1 per day.

Condition : In this mode you can write an equation which, when evaluated to true, will trigger than transition. For instance if we had a stock, [**Infection Level**] in our agent indicating how sick they were, we could have them transition out of the sick state once that stock fell to zero. The condition to enable this would be something like: `[Infection Level] = 0`.

A State Transition Diagram for Disease

This model illustrates the use of state transition diagrams to model a simple disease. This is a disease such as the flu where immunity is obtained once the individual recovers.

1. Create a new **State** named [**Healthy**].
2. Create a new **State** named [**Infected**].
3. Create a new **State** named [**Recovered**].

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

5. States represent the condition someone is in. So in our model a person can either be healthy, infected, or recovered from the infection. Now, lets add transitions that move a person from state to state.

6. Create a new **Transition** going from the primitive [**Healthy**] to the primitive [**Infected**]. Name that transition [**Infection**].

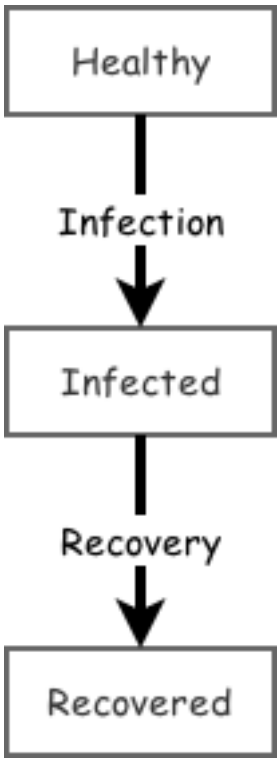
7. Create a new **Transition** going from the primitive [**Infected**] to the primitive [**Recovered**]. Name that transition [**Recovery**].

8. Please note that in this model someone who is recovered cannot become sick again. They have gained immunity to the disease.

9. Now that the model structure has been designed, let's add equations and configure the primitives.

10. Change the **Start Active** property of the primitive [**Healthy**] to `True`.

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



12. When a state is active, it means a person is in that state. By setting [**Healthy**] to start **true**, we have the person start in the healthy state.

13. Change the **Trigger Type** property of the primitive [**Infection**] to `Probability`.

14. Change the **Value/Equation** property of the primitive [**Infection**] to `0.3`.

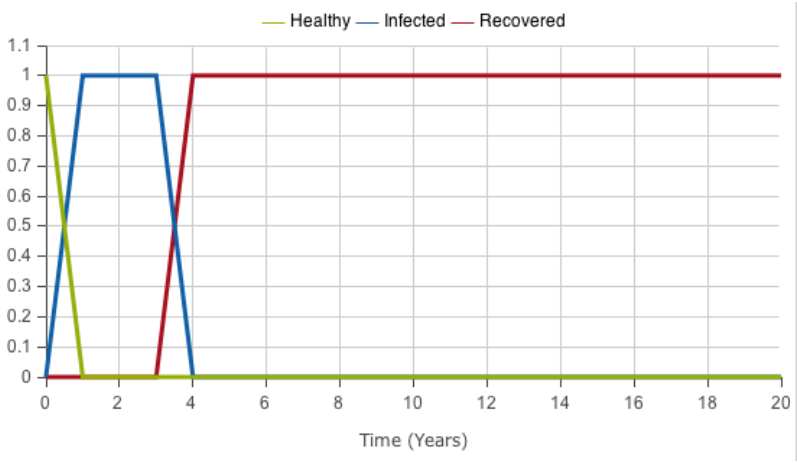
15. Change the **Trigger Type** property of the primitive [**Recovery**] to `Probability`.

16. Change the **Value/Equation** property of the primitive [**Recovery**] to `0.2`.

17. Using the Probability type for the transition trigger means that the person has a fixed probability of transitioning from one state to the next each year. Will give a 30

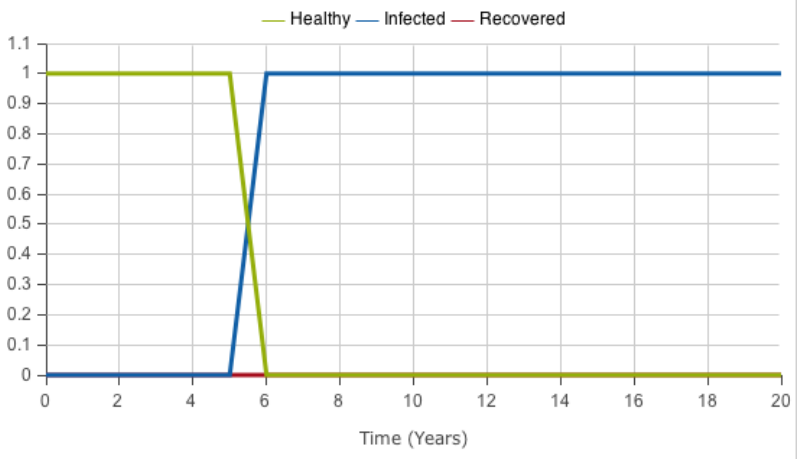
18. Let's run the model now.

19. Run the model. Here are sample results:



20. We can run it again and we will see we get different results each time. This is because the model is stochastic and the transition triggers are random.

21. Run the model. Here are sample results:



Creating Agents

You can split up the task of creating a group of agents into two steps:

1. Defining what an agent is
2. Creating a group of agents

3. Viewing results

Defining Agents

We have already introduced the folder primitive as a tool for grouping primitives together and also as a tool for unfolding a model. The folder primitive plays an important role in agent based modeling as the folder is what we use to define what an agent is.

To create an agent construct your state transition diagram for your agent (and add any stocks, flows or other variables you want). Then create a folder around all these primitives. Give the folder the name of your agent such as "Person" or "Individual" or even just "Agent". This is all similar to what we have done with folders before, but now you also need to edit the folder configuration and set the folder **Behavior** to "Agent". You have now created an agent definition!

You can have as many different types of agents in your model as you would like.

Creating a Population of Agents

After you have defined an agent in your model, you are ready to create a group or population of agents. This is done by adding an Agent Population primitive to your model.

There are a number of different settings for the agent population primitive but two of the are of key importance. The first is to select what type of agent will be in the population. Each population primitive can only have one type of agent in it. You can have multiple populations though and the agents in one population can interact with the agents in another population.

After specifying what type of agent is in the population, you need to specify how many agents are in the population at the start of the simulation. Later on you can add or remove agents to a population by using the `add()` and `remove()` functions.

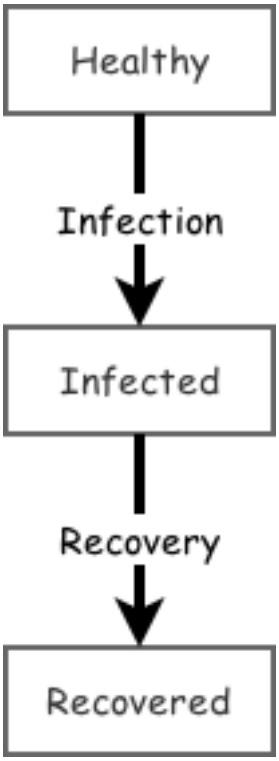
Viewing Agent Results

Many of the standard Insight Maker display types can be used to show the results of an agent based simulation. If you add an agent population to a time series or tabular display, the results for the number of agents in each state will automatically be shown. You can also use the map display type to illustrate agents.

An Agent Based Model of Disease

Here we convert a state transition diagram into a model containing multiple agents.

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



2. We start with our state transition diagram from our previous example.

3. Create a new **Folder** named [**Person**]. The folder should surround the primitives [**Healthy**], [**Infected**], [**Recovered**], [**Infection**] and [**Recovery**].

4. Change the **Type** property of the primitive [**Person**] to `Agent`.

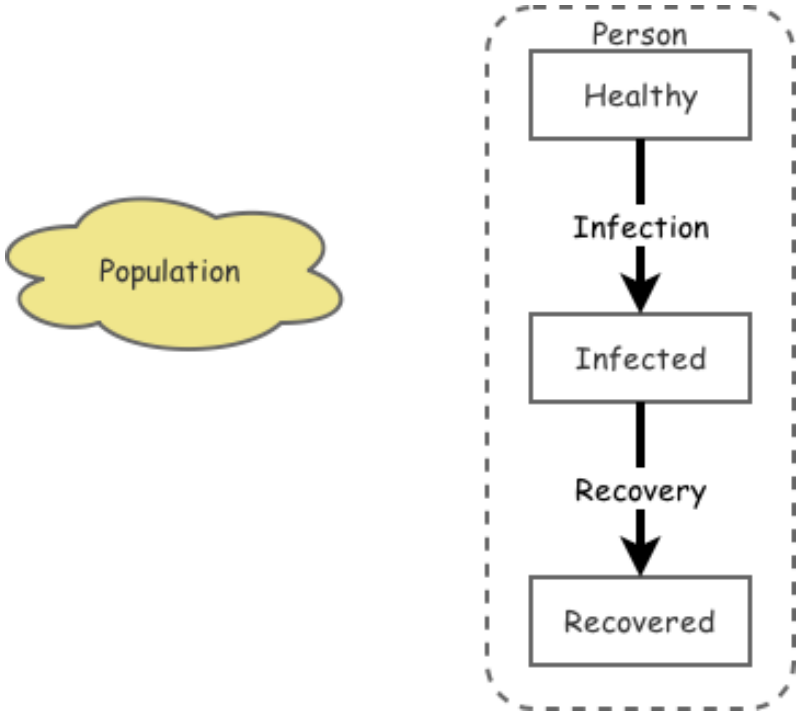
5. First we create an agent folder to encapsulate our state transition diagram.

6. Create a new **Agent Population** named [**Population**].

7. Change the **Agent Base** property of the primitive [**Population**] to `Person`.

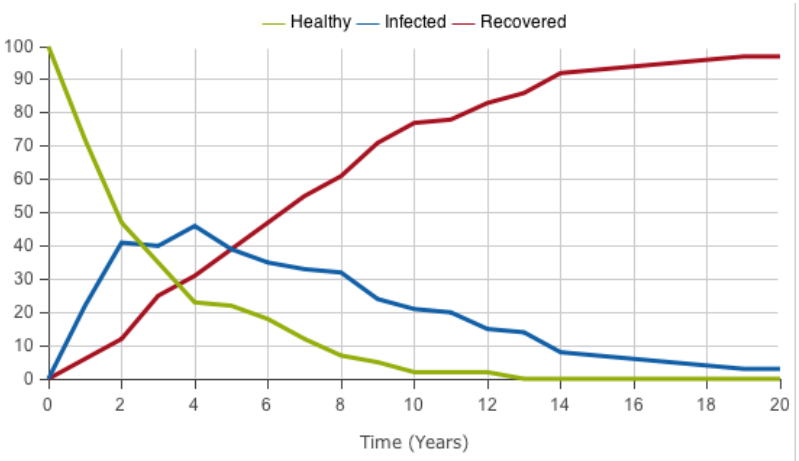
8. Next we create an agent population [**Population**] and set it to contain instances of our **Person** agent.

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



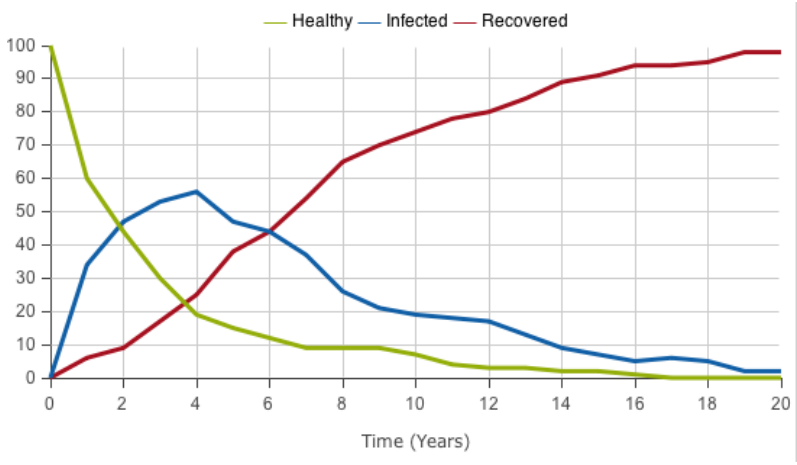
10. We can now run the model to see how the disease affects the 100 people in our population.

11. Run the model. Here are sample results:



12. Each time we run the model we will get different results due to the stochasticity in the model.

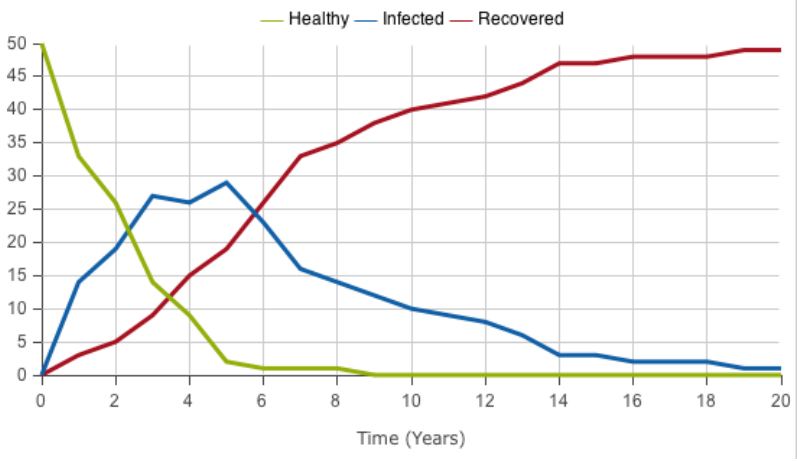
13. Run the model. Here are sample results:



14. Change the **Size** property of the primitive [**Population**] to `50`.

15. We can easily change the number of people in the population. Let's set it to 50 then run the model again.

16. Run the model. Here are sample results:



17. When we have a smaller number of people, the population trends are more variable. As we add more and more people, the trends become smoother and approach the average results we would see if we had a System Dynamics model.

Working with Agents

Working with agents is fundamentally slightly different from working standard primitives in your model. For instance if you have a regular model and refer to the value of a variable or stock you get a single value back. With agents, however, you might have hundreds or thousands of different values of a variable or stock in your model one for each of the agents that have been created.

You will need a new toolkit to be able to effectively access and manipulate this form of data into a manner that is useful for you and allows you to accomplish your goals in your model. The key building block of this toolkit is the vector[2]. In the following section we will first introduce the general concept of vectors and then show you can use them to interact with agents.

Working with Vectors

A vector is an ordered list of items. In Insight Maker vectors can written using the '«' sign (or '<<') followed by the '»' sign (or '>>'). For instance the following is a vector of the first four numbers:

```
«1, 2, 3, 4»
```

Insight Maker has extensive functions to manipulate vectors. For instance you can do multiplication using vectors:

```
2*«1, 2, 3, 4» # = «2, 4, 6, 8»
```

```
«1, 2, 1, 2»*«1, 2, 3, 4» # = «1, 4, 3, 8»
```

Or addition using vectors:

```
2+«1, 2, 3, 4» # = «3, 4, 5, 6»
```

```
«1, 2, 1, 2»+«1, 2, 3, 4» # = «2, 4, 4, 6»
```

There are also a large number of functions you can use to summarize vectors. For instance, you can find the average value in a vector:

```
mean(«1, 2, 3, 4») # = 2.5
```

Or the number of elements in a vector:

```
count(«1, 2, 3, 4») # = 4
```

There are also functions you can use to change the ordering of elements in a vector:

```
reverse(«1, 2, 3, 4») # = «4, 3, 2, 1»
```

```
sort(«3, 4, 2, 1») # = «1, 2, 3, 4»
```

A couple functions are available to combine vectors together:

[2] In some other languages vectors are sometimes called "Arrays" or "Lists".

```
Union(«1, 2 ,3», «2, 3 ,4») # = «1, 2, 3, 4»

Intersection(«1, 2 ,3», «2, 3 ,4») # = «2, 3»

Difference(«1, 2 ,3», «2, 3 ,4») # = «1, 4»
```

And lastly there are two very powerful functions we should mention: `map()` and `filter()`. Map takes each element in a vector and applies some transformation to it and returns a vector of the results. For instance, if we want to take the square of each element in a vector we could run the following:

```
Map(«1, 2, 3, 4», x^2) # = «1, 4, 9, 16»
```

Here the function `x^2` is applied to each element in the vector (with `x` representing the element value) and the results are returned. Filter takes a function and applies it to each element in a vector. If the function evaluates to true, the element is included in the results vector, if it evaluates to false, the element is not included in the results. For instance, the following function filters out all elements that are smaller than 2:

```
Filter(«1, 2, 3, 4», x >= 2) # = «2, 3, 4»
```

There are many more vector functions available, but these are some of the key ones. They will prove invaluable when you come to working with collections of individual agents.

Accessing Agents

Insight Maker includes a number of functions to access the individual agents within a population. The simplest of these is these is the `FindAll()` function. Given a agent population primitive let's call [**Population**], the FindAll function returns a vector containing all the agents within that agent population:

```
FindAll([Population])
```

So if you agent population currently had 100 agents in it, this would return a vector with 100 elements where the first element referred to the first agent, the second element referred to the second agent and so on. It is important to note that these elements are agent references, not numbers. So you can use function like `Reverse()` on the resulting vector, but you cannot directly use functions like `Mean()` and the "average of agents" does not make any sense. Will see how to access the values for agents next.

In addition to the FindAll function, there are other find functions that return just a subset of the agents in the model. For instance, the `FindState()` and `FindNotState()` functions return, respectively, agents that have the given state active or not active. For instance, if our agents had a state called [**Smoker**] that represented if they were smokers or not we could contain a vector of the agents that were smokers using the following:

```
FindState([Population], [Smoker])
```

And we could obtain a vector of the agents that were not smoker with:

```
FindNotState([Population], [Smoker])
```

The find functions can also be nested. For instance, if we wanted a vector of all male smokers, we could do something like the following:

```
FindState(FindState([Population], [Smoker]), [Male])
```

Nesting find statements is like using Boolean AND logic (like you might use on a search engine: "Smoker AND Male"). To do or logic (e.g. "Male OR Smoker") and return all the agents that are either a smoker or a man (or both), you can use the Union function to merge to vectors:

```
Union(FindState([Population], [Smoker]), FindState([Population],
[Male]))
```

Agent Values

Once you have a vector of agents, you can access the values of the primitives in those agents using the `Value()` and `SetValue()` functions.

The Value function takes two arguments: a vector of agents and the primitive for which you want the value. It then returns the value of that primitive in each of the agents as a vector. For instance, lets say are agents have a primitive named [**Age**] (it could be either a stock or a variable). We could get a vector of the age of all the people in the model like so:

```
Value(FindState([Population]), [Age])
```

A vector of ages by itself is generally of not too much use. Often we will want to summarize it, for instance by finding the average age:

```
Mean(Value(FindState([Population]), [Age]))
```

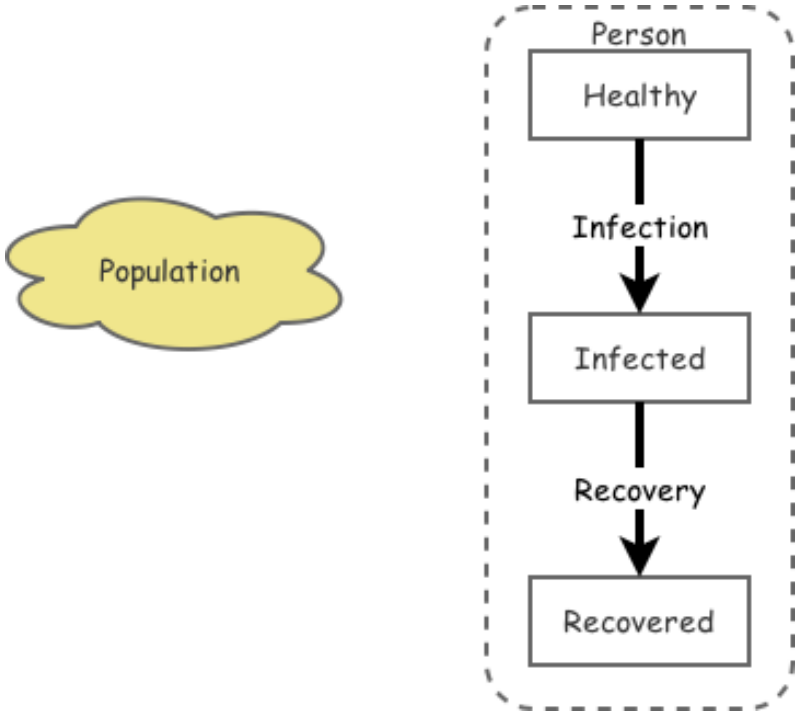
In addition to determining the value of of a primitive in an agent, you can also manually set them using the SetValue function. It takes the same arguments as the the Value function in addition to the value you want to set the agents to. For instance, we could use the following to set the age of all our agents to 25:

```
SetValue(FindState([Population]), [Age], 25)
```

Agents Interacting

This example shows how agents can interact with each other.

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



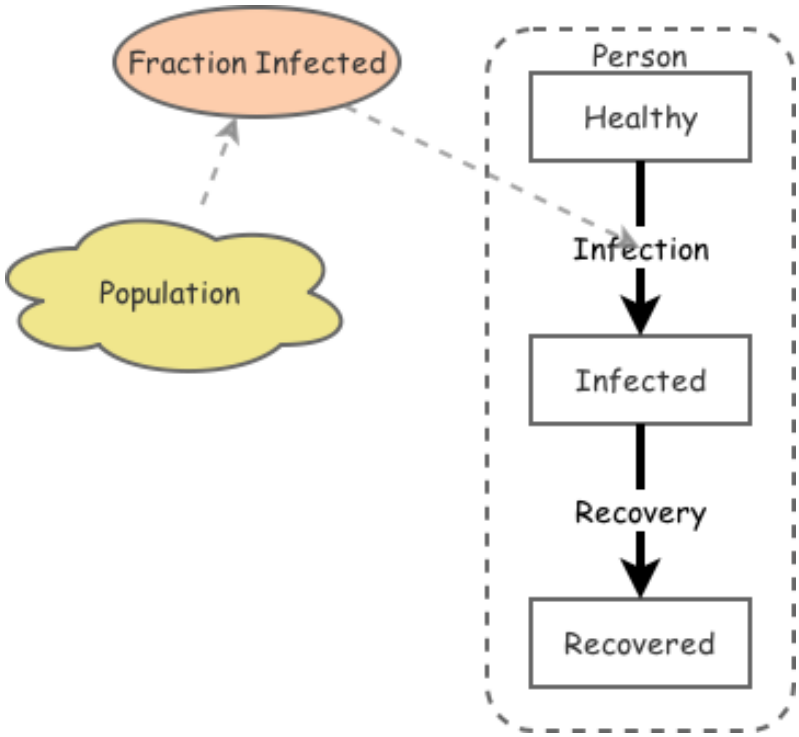
2. We're going to start with the agent based disease model from our earlier example. We are going to add a variable [**Fraction Infected**] that calculates what fraction of the population is currently infected. We will then use this variable to determine the infection rate.

3. Create a new **Variable** named [**Fraction Infected**].

4. Create a new **Link** going from the primitive [**Population**] to the primitive [**Fraction Infected**].

5. Create a new **Link** going from the primitive [**Fraction Infected**] to the primitive [**Infection**].

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



7. Now let's configure the value of Percent Infected and change the Infection transition to use it.

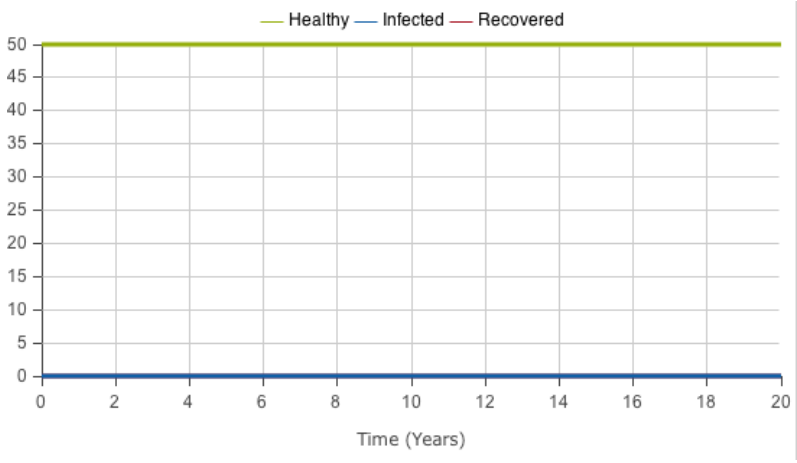
8. Change the **Equation** property of the primitive [**Fraction Infected**] to `Count(FindState([Population], [Infected]))/PopulationSize([Population])`.

9. This equation uses the FindState function to select all the people in the Population primitive who are in the Infected state. It then divides the count of those people by the total size of the population.

10. Change the **Value/Equation** property of the primitive [**Infection**] to `[Fraction Infected]`.

11. We are now ready to run the model.

12. Run the model. Here are sample results:



13. That was a bit of disappointment wasn't it? Nothing happened. Why is this?

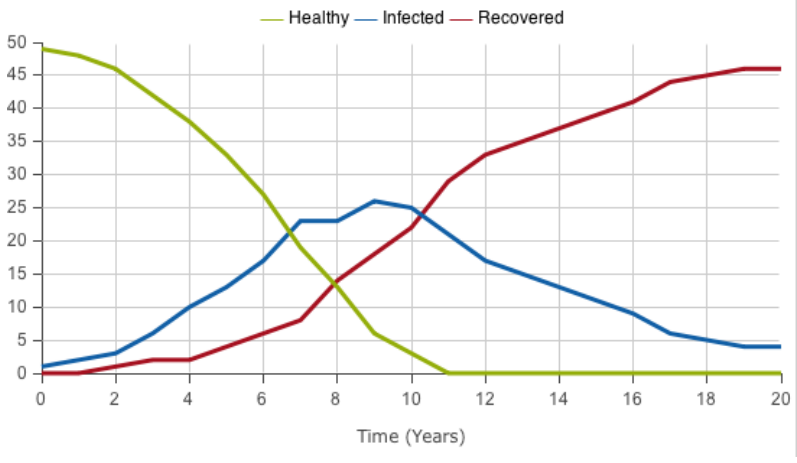
14. Well since our infection rate now depends on the number of people we have to have at least one person infected to get the epidemic going. Let's change the [**Healthy**] and [**Infected**] states so one person starts in the infected state at the beginning of the simulation.

15. Change the **Start Active** property of the primitive [**Healthy**] to `Index([Self]) <> 1`.

16. Change the **Start Active** property of the primitive [**Infected**] to `Index([Self]) == 1`.

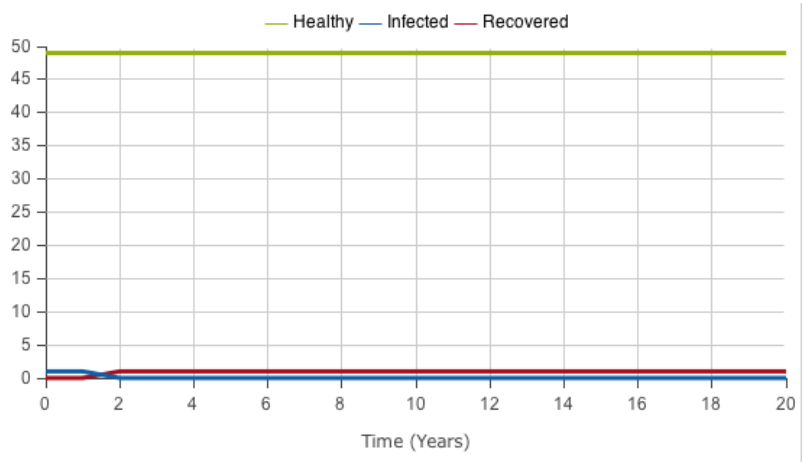
17. Each agent has an index, we have set it so the first agent will start the simulation in the infected state. Let's run the model to see this working.

18. Run the model. Here are sample results:



19. Each time we run it we will get a different set of results. Sometimes the infection will die off with just the first infected person.

20. Run the model. Here are sample results:



Agent Geography

One of the key strengths of agent based modeling is that it allows us to represent the geography between our agents. So if we are developing a disease model we do not have to assume that all the agents are perfectly mixed together like atoms in a gas. Instead, using agent based modeling we can explicitly define the physical relationship between the different agents and study how this geography affects the spread of the disease.

In general when we talk about geography we are talking about spatial geography. The locations of people with a region in terms of the latitude and longitude (and sometimes their elevation). Insight Maker supports this kind of geography, but it also supports a second kind of geography: network geography. Insight Maker supports the specification of "connections" between agents. This leads to a new type of geography where you have centrally located agents (ones connected to many other agents) and agents far from the network's center (those that are unconnected or just connected to a very few other agents).

Both these types of geographies can be useful in exploring real world systems. We will introduce them both here.

Spatial Geography

In insight Maker, each Agent Population can be given dimensions in terms of a width and a height. By default, agents are placed at a random location within

this region. You can, however, choose a different placement method for the starting position of the agents. The following placement methods are available:

Random : The default. Agents are placed and random positions with the geometry specified for the agent population.

Grid : Agents are aligned in a grid like form. You need to ensure that you have enough agents so that the grid is complete and you might have to experiment with increasing or decreasing the numbers of agents to make the grid fit perfectly for a given region dimension.

Ellipse : Agents are arrange in a single ellipse in the region. If the region geometry is a square, than the agents will be arranged in a circle.

Network : Assuming connections between agents have been specified, the agents will be arranged in order to create a pleasing layout of the network structure.

Custom Function : Here you can specify a custom function to control the layout of the agents. This function will be called once for each agent and should return a two element vector where the first element is the x coordinate of the agent, and the second element is the y coordinate. The primitive **[Self]** in this function will refer to the agent that is being positioned.

Spatial Find Functions

When working with a spatially explicit model, a number of additional find functions are available to you to obtain references to agents that match some spatial criteria.

`FindNearby()` is a useful function that returns a vector of agents that are within some proximity to a target agent. It takes three arguments, the agent population primitive, the agent target for which you want nearby neighbors, and a distance. All agents within the specified distance to the target agent will be returned as a vector.

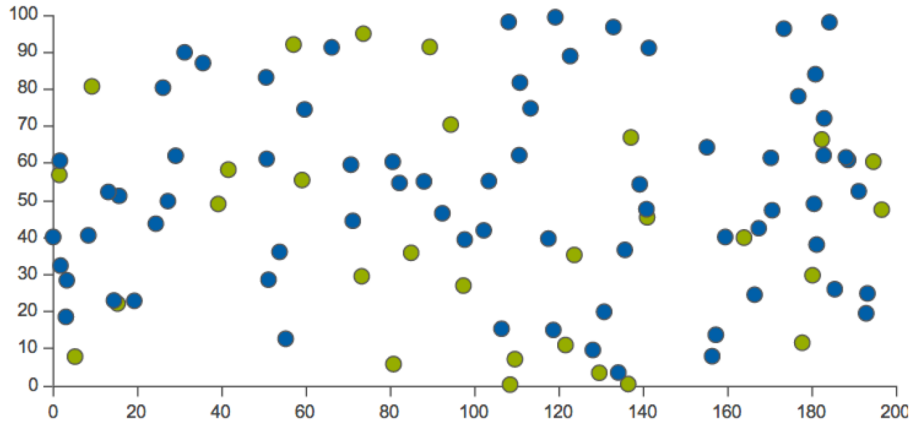
It is useful now to introduce the concept that will be very helpful to you. When used in an Agent, **[Self]** refers to the agent itself. So if you have a primitive within an agent **[Self]** can be used from that primitive to get a reference to the agent containing the primitive. So the following in an agent will return a vector of agents that are within 15 miles of the agent itself:

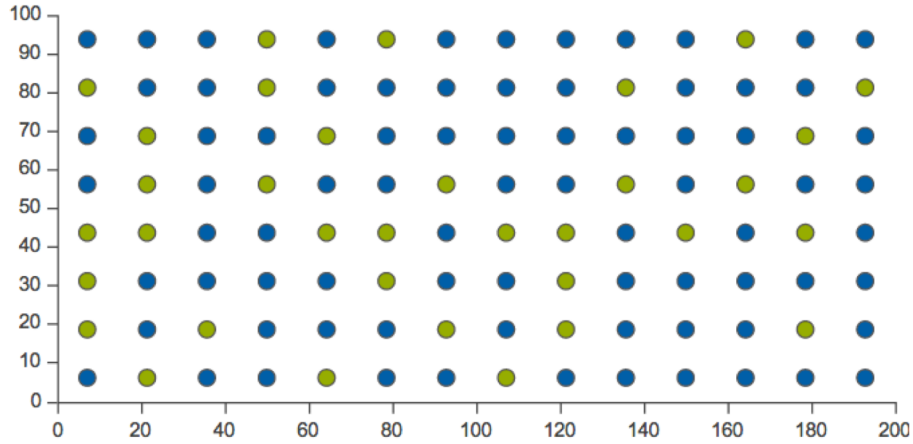
```
FindNearby([Population], [Self], {15 Miles})
```

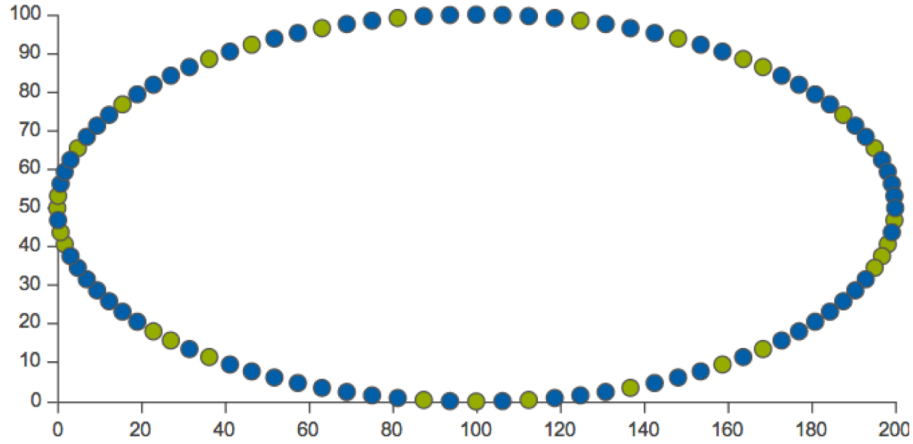
Two other useful functions for finding agents in relation to each other are `FindNearest()` and `FindFurthest()`. FindNearest returns the nearest agent to the a target while FindFurthest returns the agent that is farthest away from it. Each of them also allow an optional third argument determining how many nearby (or far away) agents to return.

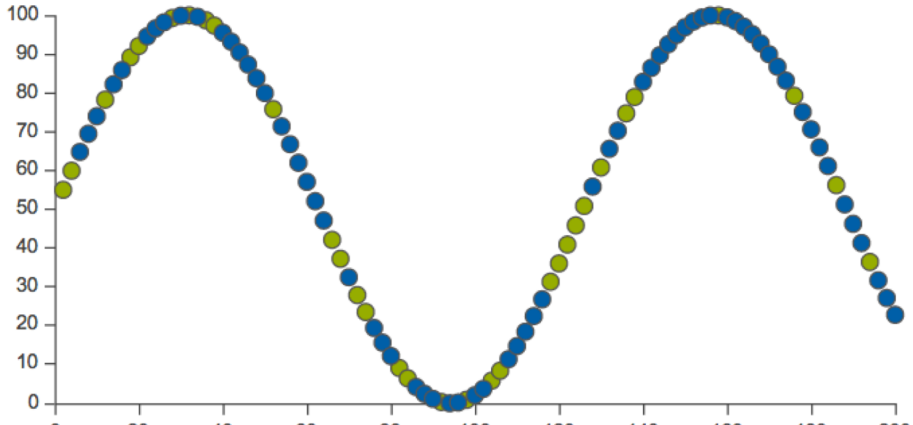
For example, the following finds the nearest agent to the current agent:

```
FindNearest([Population], [Self])
```


100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200


100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200


100
90
80
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
160
180
200


100
90
80
70
60
50
40
30
20
10
0

While this find the three agents that are furthest from it:

```
FindFurthest([Population], [Self], 3)
```

Movement Functions

You can also move your agents to new positions over the course of the simulation. To do this, it is helpful to use a new primitive we have not yet introduced. This primitive is that *Action* primitive. Action primitives are designed to execute some action that changes the state of your model. For instance, they can be used to move agents or change the values of the primitives within an agent. An action is triggered in the same way a transition is triggered. Like a transition, there are three possible methods of triggering the action: timeout, probability, and condition.

For instance, we can use and action primitive in an agent and the `Move()` function to cause our agents to move around randomly. The Move function takes two arguments: the agent to be moved, and a vector containing the x- and y-distances to move the agent. Thus, we could place an action primitive in our agent and give it the following action property to make it move the agent move randomly over time[3]:

```
Move([Self], «rand, rand»-0.5)
```

Another useful movement function is the `MoveTowards()` function. MoveTowards moves an agents towards (or away from) the location of another agent. MoveTowards takes three arguments, the agent to be moved, the target agent to move towards, and how far to move towards that agent (with negative values indicating movement away). The following command would move an agent one meter closer to its nearest neighbor in the population.

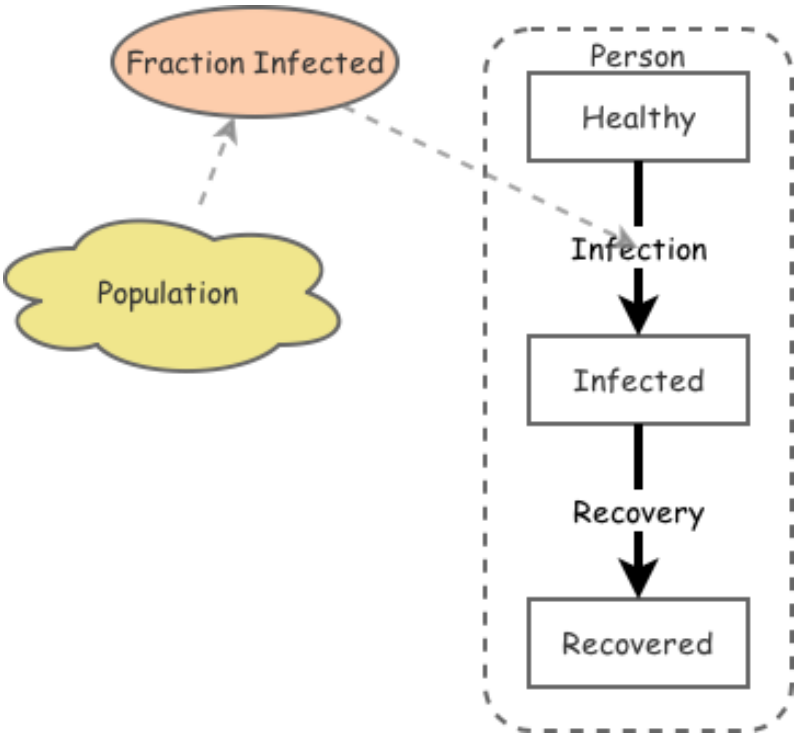
```
MoveTowards([Self], FindNearest([Population], [Self]), {1 Meter})
```

Agent Movement

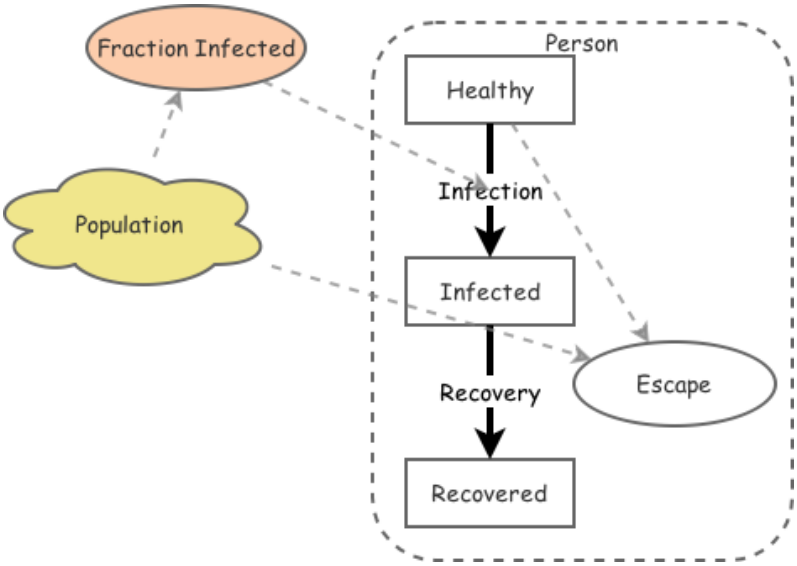
This model illustrates the use of movement within an agent based models. We adapt the model so that healthy agents run away from the nearest infected agent.

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

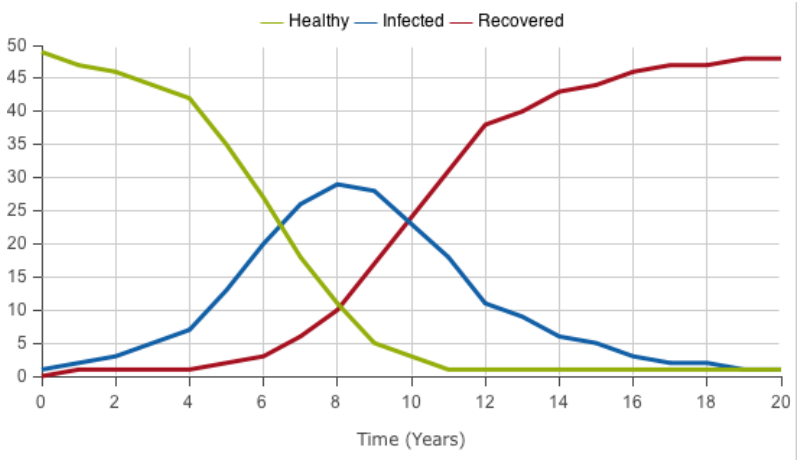
[3] What we are implementing here is known as a "random walk" or Brownian Motion. It is a commonly studied pattern of movement with wide applications in science.



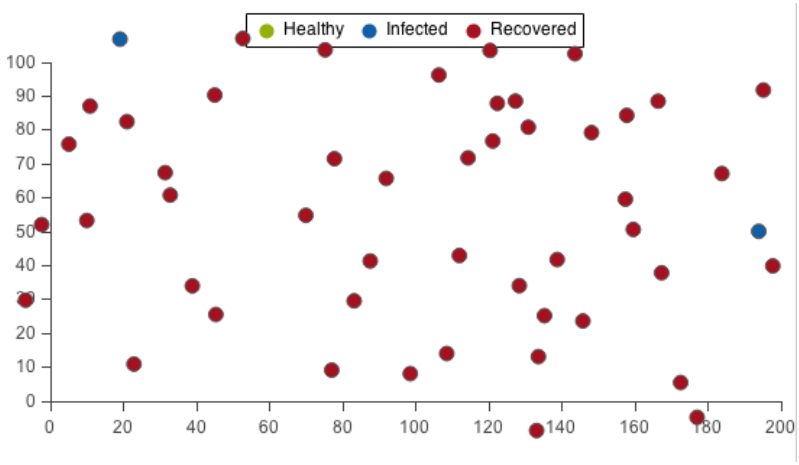
2. We will extend our model from earlier by adding movement to the agents. First we need to create an action primitive.
3. Create a new **Action** named [**Escape**].
4. Create a new **Link** going from the primitive [**Healthy**] to the primitive [**Escape**].
5. Create a new **Link** going from the primitive [**Population**] to the primitive [**Escape**].
6. The model diagram should now look something like this:



7. We will have this action be triggered when the agent it is in is healthy and there is at least on infected agent in the simulation.
8. Change the **Value/Equation** property of the primitive [**Escape**] to `[Healthy] and Count(FindState([Population], [Infected])) > 0`.
9. The action will cause healthy agents to move away from the nearest infected agent.
10. Change the **Action** property of the primitive [**Escape**] to `MoveTowards([Self], FindNearest(FindState([Population], [Infected]), [Self]), -2)`.
11. We can now run the simulation.
12. Run the model. Here are sample results:



13. These results look the same as before. To see the agents moving we need to change the display type to **Map** and then we can run the simulation again.
14. Run the model. Here are sample results:



Network Geography

To create connections and remove connections between agents you can use the `Connect()` and `Unconnect()` functions. Both of these functions take two arguments: the agents that should be connected or disconnected. For example, to connect an agent to its nearest neighbor, you could use the following

```
Connect([Self], FindNearest([Population], [Self]))
```

To disconnect an agent from its nearest neighbor (assuming they are connected), you would use:

```
Unconnect([Self], FindNearest([Population], [Self]))
```

To obtain a vector of connections to an agent, use the `Connected()` function:

```
Connected([Self])
```

By default, no connections are created when a simulation is initially started. If you change the **Network Structure** configuration property of the agent population primitive you can specify a function to create connections when the simulation is started.

This function is called once for each pair of agents in the model. The agents are available in the function as the variables *a* and *b*. If the function returns true, then the agents will start connected. If the function returns false, the agents will not be initially connected. You could use this function for, to instance, specify that 50% of agents will be directly connected to each other using:

```
RandBoolean(0.5)
```

Multiline Equations

So far in this book, the equations we have looked at have generally been straightforward mathematical equations. We have introduced some more advanced concepts, such as vectors, but for the most part our equations have been relatively simple. When doing Agent Based Modeling, at some point you will find these one-line equations to be limiting. When you start to run into these limitations with your own models you may need to start using multiline equations for certain things.

Basically, almost everywhere in Insight Maker where you can write a mathematical expression, you can also right a multiline equation. It turns out the Insight Maker's language for specifying equations is actually a flow-blown computer programming language and you can use multiline equations to implement whatever functionality you want.

We delayed introducing these capabilities as they can be a distraction from focusing on understanding a system. However, when you build complex Agent Based Models, they can be necessary to express the model logic you wish. Given this, we'll provide a brief introduction to the programming features that can be used as part of Insight Maker equations.

Variables

Variables are temporary slots to store values to be reused within your equation. Variables are created using the '<-' symbol. For instance:

```
a <- 2 # The variable 'a' holds the value 2
b <- a + 2 # The variable 'b' holds the value 4
a <- b^2 # b=4, a=16
```

Variable names can contain any number of letters and numbers, but should always start with a letter.

If-Then-Else

You should be familiar with the `IfThenElse()` function. A multiline alternative to it exists. The following is equivalent to `IfThenElse([Lake] > 10, 1, 2)`.

```
If [Lake] > 10 Then
   1
Else
   2
End If
```

One of the benefit of these multiline equations is they can be more readable than the single line functions. This is especially true if you are trying to do nested if statements. Compare `IfThenElse([Lake] > 10, 1, IfThenElse([Lake] < 5, -1, 2))` to:

```
If [Lake] > 10 Then
   1
Else If [Lake] < 5 Then
   -1
Else
   2
End If
```

The second one is much more readable. This makes it easier to maintain and more resilient to typos and bugs.

Loops

Loops are a programming construct that repeats some code multiple times. There are several different types of loops. One key loop is the *for* loop which repeats a command a specified number of times. Here is an example of it:

```
sum <- 0
For i From 1 To 3
   sum <- sum + i
End Loop
sum
```

The inner part of the loop is run three times here. The first time i is assigned the value of 1, the next time 2, and the last time 3. So this sums up the values of 1, 2, and 3 resulting in 6.

Another variant of the *for* loop is the *for-in* loop. This uses a vector to assign the number of iterations. The following code sums the numbers 1, 5, and 10 to get 16.

```
sum <- 0
For i In «1, 5, 10»
   sum <- sum + i
End Loop
sum
```

These loops can be very useful to iterate through a vector of agents. Another loop exists called the *while* loop that repeats some code until a condition becomes true. Here is an example:

```
total <- 2
While total < 100
   total <- total^2
End Loop
total
```

This code keeps squaring the total variable until the total is greater than 100. This will results in 256.

Functions

Functions allow you to reuse code in multiple places in your model. For instance, imagine you had a model that dealt with temperatures in both Degrees Fahrenheit and Celsius. Every time you want to convert from one form to the other you would have to include the standard conversion formula in your equations. Not only would this be tedious, it would also be error prone as you could make a mistake one of the times you typed it.

You can define functions two ways. One is a short one-liner:

```
FtoC(f) <- 5/9*(f+32)
```

And another is a multiline form allowing you to incorporate multiline logic:

```
Function FtoC(f)
   5/9*(f+32)
End Function
```

A great place to include your functions is in the Macros section of your model. You can enter macros by clicking the **Macros** button in the **Tools** section of your toolbars. The function you define here will be accessible in any equation in any part of your model.

Integrating SD and ABM

System dynamics modeling and agent based modeling are two different ways of approaching modeling. System dynamics looks at highly aggregated systems and encourages the study of feedback. Agent based modeling, conversely looks at individuals and the interactions between these individuals.

Although these techniques can be thought of as quite different, it important to realize that at the end of the day both of them are simply applied mathematics. To emphasize this, Insight Maker integrates both the techniques together seamlessly in its modeling environment. There is no such thing as an "Insight Maker Agent Based Model" or an "Insight Maker System Dynamics Model". There are simply models where you may use agent based techniques, system dynamics techniques or a mixture of the two techniques. Some software packages only do System Dynamics or Agent Based modeling leading to the perception that they are fundamentally incompatible methodologies.

Insight Maker (and other modeling packages such as AnyLogic http://www.anylogic.com/) allows you to integrate the two seamlessly together. For instance, in this chapter we have used state transition diagrams within our agents. We could just as well use stock and flow diagrams within our agents so that each agent in effect contains its own System Dynamics model of its state. Similarly you can have a large System Dynamics model, and have an agent based sub-model that feeds into the overreaching models.

When doing modeling its important to not get focused on labels or modeling taxonomies. Given a modeling task you want to think about what tools techniques are best used to approach it. You do not want to approach a modeling problem trying to figure out how to fit the task into your favorite modeling paradigm.

Chapter 13

References