# Module 1

## Lesson 1: Getting to Know You

In this first lesson, Getting to Know You, we'll go over the course info and objectives:

1. Identify simulation models and recognize simulation studies, and to figure out how to use them.
2. Illustrate organization of simulation languages, including Modeling with Arena, a comprehensive simulation package with animation capabilities, almost like a computer game.
3. Analyze statistical aspects of simulations including input analysis, random variate generation, output analysis, and variance reduction techniques.

Prerequisites: You should know some probability and statistics at the level of our ISyE 2027 and 2028 courses and should have stochastic processes in your background. You should also be familiar with some programming language, and maybe even a spreadsheet package.

### Suggested Resources

“Simulation Modeling and Analysis (5th Edition)" by Law, A. M.. It is the 5th edition from 2015, but you can actually get any of the earlier editions pretty much just as well.

“Arena” (6th edition) by Kelton, Sadowski, and Zupick, which concentrates primarily on the simulation modeling language called Arena.

Arena software that you can get on this website that we've listed below. It is a very nice product. It's got some limitations because it is the student version. But for all the work that we'll do in this class, it is sufficient.

### Grading (Updated Fall 2021)

We're going to have three tests. Midterms 1 & 2 are each 25% and the Final are each 30%. The remaining consists of homework 10%, and a project 10%, and whether I like you. Homework will be assigned after every module, so we'll have at least 10 assignments in all by the time the course is over. As I said before, I'm going to provide extensive course notes on the website. Now, that does not mean that you can just go around and print out the notes and skip class. You really should pay attention to what we go over, because sometimes I give out little clues that might not be in the course notes themselves. Always a good idea to watch everything so you don't miss anything.

### Programming Requirements

The course will have a lot of programming, in this Arena simulation language, also in some spreadsheet packages. But you do have some flexibility. In any case, you can expect to use the following general types of languages:

- A spreadsheet package. Everybody knows Excel these days. I'm going to have some spreadsheet add-ons that we'll talk about when we need them. They'll be widely available, not too hard to use.
- I'd like you to be able to use a real programming language, maybe like Matlab or Python, or whatever your favorite language happens to be. I'm not going to mandate the language that you use. But just have some familiarity in programming.
- A simulation language like Arena. If for some reason you don't want to use Arena, that is actually no problem. But you'll be expected to do the assignments in whatever your favorite language happens to be.

# Module 1

## Lesson 2: Syllabus

Generally speaking, the course will have certain lessons that emphasize math and statistical issues, and then certain lessons that are mostly modeling and programming of a huge variety of different systems. We kinda break them up into those two pieces.

Let's Go: I will give some introductory material on the history of simulation, followed by calculus, probability, and statistics boot camps (Law, Chapter 4). Hand simulations and spreadsheet simulations where we'll do some elementary simulations by hand or using a spreadsheet package. General high-level modeling concepts (Law, 1&2). What is a simulation? What kinds of ways can you go about modeling different systems? That material is covered in Law's book, chapters one and two. A short discussion on verification and validation (Law, 5), used to determine whether the simulation is doing what you think it ought to be doing. (Law devotes an entire chapter to that topic in his chapter five.

Arena Fun: Then I'll start work in the Arena simulation language as a large aside. I'll go over some Arena basics (KSZ, Chapter 4; though I will sneak in some Arena examples before this). We will do a generic call center in Arena, and then look at some inventory modeling techniques also in chapter five of Kelton et al (KSZ, 5). Then we'll go over a manufacturing center (KSZ, 6) which is very, very interesting because it corresponds to very, very, very general examples that don't apply just to manufacturing centers. Then we'll talk about entity transfers in Arena (KSZ, 7). What that means is how do things move around in Arena? How do you get from place to place? And finally, we'll do some advanced material in Arena (KSZ, 8), a whole potpourri of different topics that I found useful in the past. Then, back to the math stats probability aspects of the course, we'll talk about random number generation.

Randomness: How do you generate randomness on a computer? Now believe it or not, randomness is not a thing that you get for free on a computer (Law, 7). In fact, all randomness is fake. It is a big lie that they're telling you. But what I'm going to do is I'm going to show you how to generate randomness that looks random, not quite random, but it looks random to you and me, and everybody accepts it. And then I'm going to take that work and generalize it to random variate generation (Law, 8). So, this randomness can be used to generate all sorts of interesting things, single random variables (normal and Bernoulli random variables), things that you heard about in your prob and stats classes. I'll also talk about things like multivariate random variables. If one dimension is good, two dimensions are even better. So I can generate people's heights and weights simultaneously, things that are correlated with each other. After that, I'll generalize even more and talk about random processes like time series -- for instance, a simulation of unemployment rates which are correlated from month to month, things like that. And as a special case, we'll look at certain financial models, things like stock prices, option prices, very, very, very interesting models that you can use. You can generate almost anything with a simulation. We'll show how to do some of those in this module. So that turns out to be Law, chapter eight. That is a fairly substantial chapter.

Stats Issues: Finally, I'll look at a number of additional statistics issues. Input analysis (Law, 6), what that means is what random variables should you use to drive the simulation? If the random variables you use to run a simulation aren't very good, then the whole simulation is no good. That is called garbage in, garbage out. So what you have to do is model the input random variables to the simulation correctly, and we'll talk about how you do that. What goes in, must come out, and so we'll also look at so-called output analysis (Law, 9). In other words, analyze the output coming from the simulation. It turns out even if you've taken a statistics course, everything they told you in that class was pretty much a lie when it comes to simulation, and you'll see why. The output from simulations is surprisingly difficult to analyze and you have to be very careful about that.

One of my favorite topics is comparing systems (Law, 10). One of the reasons you would run a simulation is to see if one system is better than another, or if one system is the best among many. We'll devote an entire module to that. And finally, we'll have time to do a variance reduction and other cool things. Variance reduction is basically, well, how do you run the simulation really, really, really efficiently? And there's a lot of nice little tricks associated with that. So that comes to the end of the syllabus. The summary for today was that we've chatted about the syllabus, as I promised. And next time, I'm finally going to get into some real simulation by beginning our whirlwind tour of the subject.

# Module 1

## Lesson 3: Model

We're going to start this tour with general modeling issues and why would we even consider using computer simulation, you know, what is all the fuss about?

### Model types and definitions

Models are *high-level representations of the operation of a real-world process or system*. A **discrete model** is one that changes every once in a while, at discrete points in time. For instance, a model of customer flow in a store would simulate a customer appearing and getting served, but between those events, nothing else is going on. Then you wait in line, but nothing is going on then. You're getting served, but while you're getting served, nothing is going on. So, at discrete points in time, the system changes. Somebody shows up, somebody leaves. In contrast, a **continuous model** changes constantly, like the weather.

We'll also look at **stochastic models** versus **deterministic models**. I like probabilities and statistics, that is stochastic. I'm not really as much into deterministic type things, although we will show a couple of deterministic models. Deterministic models are really boring, because nothing is happening there. It is a little bit of a joke, but stochastic models are much sexier.

Then I'll also look at **dynamic** versus **static models**. A dynamic model changes. A static model is the same thing over and over and over again. So again, dynamic models are much more interesting to me. So with that in mind, I'll say most of the time we're going to be looking at discrete, stochastic and dynamic models, not quite all the time, but most.

Now, how can you solve a model? Generally speaking, at a high level, there's three ways to do this. You could use **analytical** methods. Analytical method is like solving an equation, you get an answer, integral of x dx is x square over two plus c, that is an answer, that is a nice equation and you know, if you're lucky, you'll be able to come up with an analytical method, an analytical solution to your particular model. Usually those are pretty easy and I'll give you an example in the next slide. If an analytical method doesn't work, maybe that is when you go to **numerical**. A numerical method might be used to solve an integral, like e to the minus x square. It turns out that thing doesn't have a closed form and you might need to use numerical integration methods, something of that ilk to solve those kinds of problems. So those are pretty good. Generally speaking, numerical methods work well. You can use those if you can't use analytic methods.

Now what happens if you can't use a numerical method? Well, then you may have to resort to **simulation**. Look at that, it is so important we colored it in blue. So, simulation methods are way more general than these other things and they can usually help you solve methods that are too complicated for these other two, analytical or numerical.

Here's some examples of models. I can toss a stone off of a cliff and you can model the position of that stone via the usual physics equations that you took back when you were a freshman. Those have equations, those are analytical models and you just look them up, very, very simple. Now, modeling the weather, that might be a little bit too tough to use exact analytical methods, so you might have to use numerical methods and in fact, in this case, you may require numerical methods to solve a series of hundreds of thousands of partial differential equations. That is how they model the weather sometimes. Certainly, there are no closed form answers for tough things like that. Now on the other hand, if I add a little bit of randomness into that weather problem, all of a sudden, even the numerical methods might not work. I can make a simple model very difficult by adding randomness in a certain way and at that point, your kind of forced to use simulation. Now, I'm going to give you plenty of examples coming up in the next few lessons, so simulations just got so many uses, you can't believe it, but anything that you can't do with analytical or numerical methods, simulation might be a very good alternative.

So, what is simulation? It is just the imitation of some real-world, or even imaginary process over time. The trick involves generating some kind of history of the system and you use the history to get data on the system to draw inferences concerning how it operates. Even if the system isn't real, you can use a conceptual system, simulate that and make guesses of its future performance. Simulation, it turns out, is one of the top three industrial engineering/ operations research/ management science technologies. I think the other two, probably statistics and engineering econ, but simulation is the best one, take my word for it. Now, it is used by academics and practitioners on this huge variety of theoretical and applied problems, so most people use simulation on real-world, applied problems, but people use simulation all the time on theory-oriented problems as well. In any case, it is an indispensable, problem-solving methodology, that everybody could use quite successfully.

## Module 1

So, what is simulation good for? Well, you can use simulations to describe, analyze real system behavior or even conceptual system behavior, as I mentioned before. You can ask what if questions about these types of systems, like, what if I added another server, how would that improve my queue waiting time? Simulation helps a great deal in system design and optimization. Some aircraft companies use simulation before they even think about building a new aircraft. Plus, I can simulate almost anything. I can do customer-based systems like manufacturing systems, supply chains, health systems, countless applications and I can also simulate systems, that don't really have customers, as we would think of them. I could simulate stock prices; stock option prices and people use simulation for that all the time and there's plenty of reasons to simulate.

You might be interested in whether or not a particular system can accomplish its goals. Are you going to be able to get an order out on time? Will the system capacity be enough to get all the materials through the system in a timely way? What if the current system doesn't accomplish the goals? What do you do? Do you add another server? Do you add buffer space? What if the system needs just a little bit of improvement? What things can you tweak to improve the system a bit? You can also use simulation to create a game plan. What action should you take next? You can base different strategies on the outcomes of simulations of those different strategies. What if you have a known problem in your system, like a bottleneck? Well, you can simulate what would happen if you added another server at the bottleneck, but maybe the bottleneck moves downstream. The simulation will help you identify things like that. You can resolve disputes. Suppose two people at the company are having a fight over different strategies, that you could adapt in a particular problem. Simulation might help you to figure out which strategy is actually better. You could also use simulation to sell an idea. Simulation is so easy to use, at least for certain problems and it also has the potential to be very easy to understand and you can use it to sell your ideas.

In fact, simulation has got a lot of advantages. You can study models, that are far too complicated for analytical or numerical treatments. You can study very detailed relations, that might be lost in those kinds of treatments. For instance, an analytical treatment might be able to give you the exact, expected waiting time in a queue, but it doesn't necessarily tell you what percentage of customers are going to have to wait a certain amount of time or what percentage of customers get mad and leave the system. The simulation can give you many, many more details than just an "answer". You can also use simulations as the basis for experimental studies of systems. You can run the simulation beforehand and then it'll give you an idea of how many runs you have to take, maybe with real data. The simulation is very useful for that. You can use simulation to check results, that other people have obtained by other methods. So, if your pal has done a little queuing theory study and says, "I believe we need four servers at this station," then you can run a simulation of that station, see if four servers is actually enough. Hopefully, you don't embarrass your friend. You could also use simulation to reduce design blunders. There are countless examples, where if they had simulated the darn system beforehand, then they would have seen, "Oh, no bottleneck here," or, "We should have put more space there," and simulation is very easy, a very easy method to avoid those kinds of issues and like we've said before, it is a really nice demo method. And finally, it turns out that sometimes, if you're a little bit lucky, simulation is very easy to do, and it is actually kind of fun. It is like a video game almost.

Now, I have to be honest, there are also some disadvantages. I hope the advantages outweigh the disadvantages, but let's go over a few of those. Sometimes simulation is not so easy, I mean, it is not a panacea. Sometimes you actually have to roll up the sleeves and do a lot of programming. Sometimes it is very difficult to collect the data, that you need from the simulations. You gotta be a little careful, but simulation is often a little harder, than you might think. Sometimes it is very time-consuming, very costly, because all programming is, it is just a matter of course. Now, the more subtle thing is that simulations don't give you "the answer". They typically give you random output (and lots of misinterpretation of results is possible), just like you would get if you were looking at data. The data that you get is random, in a sense, so simulations don't give you the answer. What you have to do is you give whatever this answer is from the simulation and then you have to attach confidence in it, like I think that the queuing, the average queuing time in the system, a person is going to have to wait five minutes, plus or minus 30 seconds. You always have to give that plus or minus. I mean, we're really statisticians here and you have to be careful about that. So, we'll talk about simulation output a lot later on, but you have to be careful. Remember I told you, everything they taught you in Stats class was a lie and that really does hold. In fact, simulation is a great problem-solving technique, but to do certain problems, better methods may exist. If I'm just tossing the stone off the cliff, and I'm assuming it is not raining or anything and there is no air pressure issues and no friction, then your equation from Physics class might be just fine, so you won't need simulation, but I've found that a lot of times, simulation is in fact the best method.

# Module 1

So, here is the summary of what we just did. I finally started this little tour about the nature of simulation and simulation models and next time, I'm going to start talking about the history of simulation, so I'll give this historical, or hysterical presentation.

## Knowledge Check 1.3

1. We are interested in modeling the arrival and service process at the local McBurger Queen burger joint. Customers come in every once in a while, stand in line, eventually get served, and off they go. Generally speaking, what kind of model are we talking about here? (More than one answer below may be right.)

A. Discrete
B. Continuous
C. Stochastic
D. Deterministic

2. Which of the following can be regarded as advantages of simulation? (More than one answer below may be right.)

A. Simulation enables you to study models too complicated for analytical or numerical treatment.
B. Simulations can serve as very pretty demos that even University of Georgia graduates can understand.
C. Simulation can be used to study detailed relations that might be lost in an analytical or numerical treatment.
D. Simulations are often tedious and time-consuming to produce.

# Module 1

## Lesson 4: History

Simulation has really come into its own in the last 50 or 60 years with the rise of computers. But it has been around a very long time -- at least a couple hundred years.

### Early Simulation

In 1777, Count Buffon drew some parallel lines in the ground, threw a needle onto the ground and counted up the number of times that the needle intersected one of the lines. From there, he was able to back engineer an estimate for pi. And the more times he performed the experiment, generally speaking, the better the estimate of pi. While not the best way to estimate pi, it serves as an early example of how simulations could be performed, sequentially at least.

In the early 1900s, William Gosset, was working as a statistician at the Guinness Brewery in Ireland. Guinness didn't want the world to know that it was employing a statistician because that would give away some proprietary information that they were doing certain types of quality control but approved of Gosset anonymously publishing some of his results in the statistical literature. Calling himself "Student," Gosset published some of his results on what is now known as the Student t distribution that he obtained when he was at Guinness. In order to obtain Student's t distribution, Gosset took several hundred samples of data, tossed all the samples in a hat, and then picked them out randomly. (The data was how long were the index fingers of British prisoners.) We actually still use the Student t distribution today in every stats class.

### Computer Simulation

The first use of computer simulation occurred after World War Two, when mathematicians Stan Ulam and Johnny von Neumann (1946) simulated thermonuclear chain reactions in the context of the development of the hydrogen bomb. The 1950s and 1960s also brought a lot of industrial applications of simulation, particularly in the context of manufacturing and certain queueing models. People also started to develop simulation languages. This made it easier to write programs that could handle generic problems in manufacturing and queueing. Eventually, these were developed into easy-to-use modeling tools for these generic models. They had nice graphics. You could simulate things more quickly. One early language, SIMSCRIPT, was developed by Harry Markowitz, who also won a Nobel prize for some of the work he did in optimizing financial portfolios.

In the 1960s, people actually started putting forth very rigorous theoretical work having to do with simulation. Namely, the development of very precise and efficient computational algorithms. Some very nice probabilistic and statistical methods to analyze simulation output and input. And theory started developing in a very, very strong way. We'll be looking at that as the course progresses.

### Origins: Manufacturing

Simulation began, for our purposes, with manufacturing problems. Simulation was the technique of choice because these manufacturing problems are too hard to do analytically or numerically. In particular, simulation is very easy for calculating the movement of parts and how these parts interact with system components like machines and other types of servers. Obviously, the simulation can be used to evaluate how the parts flow through the system. Simulation can be used to examine conflicting demand for resources. Now if you've got millions of parts flying through the system, and each one demands particular machines or people to work on them. Those are going to cause conflicts, and the simulation can be used to see where those conflicts are occurring and maybe even what to do about the conflicts. And in particular, when asking what to do about the conflicts, maybe you have two or three possible solutions, and you can study those contemplated solutions or changes before you actually introduce them. In particular again, you can avoid lots and lots of design blunders by simulating beforehand.

Some typical questions that might arise, especially in the context of manufacturing:

- What will be the throughput?
- How can we change it?
- Where are the bottlenecks?
- Which is the best design?
- What is the reliability of the system?
- What is the impact of breakdowns?

What's the throughput gonna be? How many parts can you get through the system? In fact, later on, as we look at more general problems How many people can get through your hospital during an emergency situation? Anything to do with throughput. How can we change the throughput? Maybe we have to add more servers. Maybe we don't need as many servers. But simulation's an easy way to figure out how we can implement changes. Particularly manufacturing systems, you're gonna encounter bottlenecks frequently. Well, where are they? And if you do something about the bottlenecks, does that cause bottlenecks to occur in other places. Simulation will answer that lickety-split.

One thing that I'm very interested in and suppose that you have to decide among five or six possible designs. You can use simulation over and over again to figure out which is gonna be the best design most of the time. And so, this is a form of optimization, simulation's very helpful with that. If you're looking at a giant network, you might wanna know what's the reliability of the network. How can you get stuff from the beginning to the end, in a reliable way, without machines failing, so as to prevent you from getting from A to B? So reliability's very important. Simulation can check into that. And finally, what's the impact of breakdowns? If a machine is to break down, what are you gonna do about it? Should you have redundancy? How long's it gonna take to get the system back up to speed, etc.?

So the summary of this lesson is that well you can kinda take your seat belts off. We went through a brief history lesson. Kinda fun. Simulation's been around a long time. It's really come into its own in the last few decades. And the next lesson, I'm gonna actually look at very specific applications of simulation. You'll see that it's very, very wide ranging and a lot of them are really interesting. So, looking forward to talking to you again.

### Knowledge Check 1.4

1. Who is William Gosset?

A. He invented the t distribution that is used ubiquitously in statistics.
B. He invented the s distribution that is used ubiquitously in statistics.
C. He invented tea.
D. He invented the word "ubiquitous".
E. He is the brother of Louis Gossett Jr., best known for his fine acting in many films, including An Officer and a Gentleman.

2. YES or NO? Has anyone closely related to the field of computer simulation ever won a Nobel Prize?

# Module 1

## Lesson 5: What Can We Do For You?

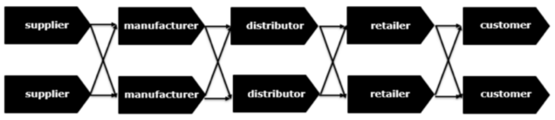
This time, we'll be looking at the kinds of stuff that simulations actually used for. Again, surprisingly wide range. And I'm gonna give away the punchline. Simulation can be used for everything. If you can describe it, you can simulate it. And we'll look at all these different applications coming up.

### Actual Applications

These are actual applications either I've done these or let's say my friends have, but these are actual applications that I've run into either by myself or with colleagues. In manufacturing, for instance, we've done simulations in various applications, in particular, automobile production facilities a number of simulations actually. Carpet production facilities, those are really big in Georgia. How does material flow through these plants? I really like doing queueing problems, and you'll see when we do the simulation package later on in the course, the language that we use, Arena, is very easy to work on queueing problems with. So, for instance, one of the sorts of bellwether examples is call center analysis. How many servers do you need in a call center in order for calls to proceed through in a timely way? Another wonderful example is modeling a fast-food drive-thru. We have a particular restaurant here near Georgia Tech where the line at the fast-food drive-thru can only be about four or five cars before the line starts extending onto a busy street. So, we simulated different strategies on getting customers through that line. What happens if you have some of the employees come out and take orders manually in the car as opposed to going through and talking to the machine? And this particular fast-food outlet, which is a chicken-related outlet here in Atlanta, has really done a good job. They simulate all of their restaurants to this day. We simulated another fast-food drive-thru with a slightly different intent. You've noticed that in this next item, I've linked that up with a call center. Let's suppose some very large fast-food restaurant with many restaurants around the country wants to have all their fast food handled through one giant call center. So, you drive up to the little machine and you talk to the person and that person isn't in the store itself. That person is in Chicago with a hundred of their best friends. And so, we wanted to know how many of the outlets did you need in order for this to be an efficient way of doing things? And it turns out, we needed about 150 restaurants before this became efficient. We've also done some work in airport security lines. I'm sure everybody encounters those. We only did the good things in the security line, the bad things that cause you to wait a long time, they were probably done by somebody else, not us. I've also done some very fun, little easy simulations of ice cream shops, other fast-food restaurants. (Combination meat processing center / fast-food restaurant / amusement park)

### Generic Supply Chain

Let's look at a generic supply chain. These are just typical examples. A supply chain might consist of a set of suppliers and then feed into manufacturers and then distributors and then retails, then customers, depending on the level of the chain. And there may be many of each and they might be interacting with each other. It is just a giant queueing network. So, any node can connect to any other node here just like a network. In fact, it looks like a flowchart: This particular chart goes from left to right, you could have these arrows go all over the place. But it is basically a flow chart.



In 1985, a famous professor and CEO of a supply chain company actually asked why would anybody want to simulate a supply chain? And it turns out that there's lots of reasons. Because back then, in your supply chain, you would just operate with means as opposed to probability distribution. So, you'd always say, well on the average, it takes five days to get from A to B in the supply chain. But averages aren't very good sometimes. In fact, there is this thing called **The Flaw of Averages.** If you use averages too much, your results are going to be non-realistic because averages don't quite take into account variability. So that is why simulation is such an important tool. In fact, many supply chain tools these days have simulation capability. So, we can use simulation to determine how much value-added a particular forecasting application provides in the supply chain. So, if we're thinking we have this certain set of forecasts, how is that going to supply what is going on, how is that going to apply to what is going on in a supply chain? Simulation is a good way to characterize that. You can also use simulation to analyze the randomness or model errors. This is the big deal in a supply chain. Is your supply chain solution robust? What is the best solution? And we'll talk more about that as the course progresses.

# Module 1

## Other Applications

Lots of inventory and supply chain analysis. Financial analysis, that's huge these days. If you can do portfolio analysis and options pricing, often with simulation, Wall Street has a job for you. If you go to your financial advisor, they brag about the portfolio analysis that they do via simulation. They try to use fancy words, but they'll say something like, "well we do five hundred Monte Carlo simulations of our proposed portfolio allocation strategy for you." And you're supposed to be impressed with that. Now you'll be able to do it yourself. We also do a lot of traffic simulation. A lot of the time, traffic flow is a little more complicated than you might think. We've discovered situations in which you add a lane, and it causes more congestion. Related to that is airspace simulation. How long does it take aircraft to get between two different destinations? That depends on what if it is really congested at Hartsfield-Jackson, or there is a thunderstorm in Atlanta. That means they're going to have to put traffic holds on some of the planes coming in from, say, New York? There's lots of activity in the service sector. Anything you can simulate very easily if you put enough time into it.

There are many healthcare applications for simulation. A classic example is modeling patient flow in a hospital. How many rooms should you allocate for different purposes in a hospital? How do you optimize doctor and nurse scheduling? You never know how many people are going to show up in the emergency room on Friday night. And if you haven't done good scheduling, maybe you're just using this Flaw of Averages, you put in the average number of doctors that you need, you could be setting yourself up for a big problem if there is a fairly major fire or if there is a car crash that you hadn't anticipated. Simulation is also used for procurement of supplies, that has to do with inventory. I've used simulation a little bit for purposes of disease surveillance. How do you know that a disease is actually in the process of occurring? This is really fascinating. I'm also really interested in my specialty areas these days, it has been in the propagation of disease spread. How does a disease go through a population? Is it a virulent disease like 1918 influenza, or something less intense like swine flu happened to be a few years ago? And a lot of what we do at GT has to do with humanitarian logistics, and we use simulation for that all the time.

Let me now talk about a specific health systems application, namely in surveillance. So, you can use simulation to monitor certain times series. That is basically what surveillance is. Is something happening in your time series? Has the unemployment rate spiked for some reason? Or has a disease started to occur for some reason? You look at a time series of data and you kind of see what happens. Is anything interesting going on? So, the idea here is to predict issues as or before they happen. Is the influenza outbreak starting to occur now? You may have read in the paper that Google is very good at predicting these types of things, sometimes in a scary way. They'll look at some of the purchases that you've made at a store, and they'll be able to determine you're pregnant. And this is maybe before you've told your family. So, these predictive analytics are one of the hottest topics now in operations research and operations management and computer science. I'm most interested in, is a disease starting out and is it in the process of becoming an outbreak? When is something occurring that is out of the ordinary? And what is nice about this, as I alluded to, is that you can take advantage of these huge data sets that are available now. Google, for instance, can look at everything that they have that all their users are doing. It is just amazing what is out there.

## Hypothesis Testing

I'm going to give a really strange surveillance application here. He is Dr. Harold Shipman, a British serial killer who used morphine and heroin overdoses to kill patients, mostly elderly women. He was caught after he carelessly revised a patient's will and left all her assets to himself. It turns out he doctored the records to show that the patients had actually needed morphine, but software recorded the dates of the modifications, and they were backdated so they were able to tell that something funny was going on.

What does this have to do with simulation? It turns out, surveillance uses what are known as sequential statistical hypothesis tests, where the hypothesis test might be something like, we usually call in $H_0$, the hypothesis might be, null hypothesis might be, no disease or in this case no murder. So that is a null hypothesis, and you have to come up with ample data to disprove that. People are innocent until proven guilty. That is the only time that you would reject the null hypothesis. So, it turns out though, that for complicated problems, these test statistics that you use to determine whether or not the null hypothesis is true, they might have very difficult, complicated statistical distributions, even if the null hypothesis, which is usually easy, even if that is true. Even under $H_0$, that is how we would say that. The test statistics might have difficult distributions. They might not be normal or exponential or the things that you learned in your baby stats course. So, the nice thing is that I can use Monte Carlo simulation to approximate the probability distributions of these very difficult statistics. Simulation can be used for lots and lots of stuff, this is one application. And then, when I take my sample and I compare it to what this distribution is supposed to be, if the sample doesn't look like it is coming from the distribution, then reject $H_0$. And this is one of the ways that they were able to catch this guy. So, simulation can be used for everything.

So here is a summary of this lesson. We looked at a wide variety of practice simulation applications. I went a little crazy on some of them. In the next lesson, what I'm going to be doing, is I'm going to be giving some really easy examples of the actual use of simulation in very simplistic settings.

## Knowledge Check 1.5

Which of the following are areas where simulation has found substantial application?

A. Inventory and Supply Chain Analysis
B. Financial Analysis
C. Manufacturing
D. Health Systems
E. Transportation Systems

Why might simulation be a good tool to analyze supply chains?

A. Supply chains are always deterministic systems.
B. Supply chains often have complicated network structures, making exact analysis difficult.
C. Supply chains are , with random travel times, lead times, and order patterns.
D. Supply chain simulations can be programmed in a matter of minutes.

# Module 1

## Lesson 6: Some Baby Examples

### Example 1: The Birthday Problem

Let's assume there are 365 days in the year, and that everyone has an equal chance of being born on any particular day. So, January 1st, probability of one over 365. April 22nd, which is my birthday, one over 365. How many people do you need to have in a room in order to have at least a 50% chance that at least two of them are going to be having the same birthday? A. 9 B. 23 C. 42 D. 183

183 is what I originally thought when I first heard this problem. In fact, if you have 183 people in the room, there's a 99.999% chance that at least two people will have the same birthday. In fact, with 42 people in the room, you have a 90 ~ 93% chance. 23 is the right answer. Let's actually simulate this time.

47475 is what is known as a random number of seed, an integer that I pick out of my head, just to get the party started. And you'll see some interesting properties of this seed. And then, I've got a little calendar here that I made. Every time I click off a person entering the room, I generate their birthday randomly, with probability of one out of 365. It turns out, in this made-up example, I only needed 24 people before I finally got a match. Let's run it again with a different seed, 12345. With each click, the program generates a new birthdate. We got a match after 19 days. It wasn't 23, the statistical 50/50 point, because every time I run the simulation, I'm going to get a different answer. This particular time, it was 19. Let's run this again, still using the seed 12345. Look at that, it was February 5th, just like before. Isn't that a coincidence? The simulation ended again, on the 19th birthday. And the reason is, *since I started with the same seed, I'm going to get the same answer*. Now, simulations are supposed to be dealing with random numbers. In fact, the numbers are not random, as this demonstrates. They just look random to you and me, which is good enough, most of the time. But if I start with the same seed, I get the same answer. It turns out, this is a good thing, and we'll talk about that in great and tedious detail later on. Let's do another seed. 17 this time. If I change the seed one more time, it only took six tries before a match.

### Example 2: Estimating pi

I'm going to use Monte Carlo simulation to estimate pi, 3.14159. The idea here is related to the 1777 Buffon's needle problem. What is the area of a unit square? I'm going to inscribe a circle inside that unit square. The area of the inscribed circle is pi r2, which is pi * 1/2, or pi/4. Therefore, if I toss a random dart at the square, the probability that it hits that inscribed circle will be the ratio of the areas, pi/4. I'm going to throw lots of darts and count out the proportion of darts that land in that circle. The proportion should approach pi/4 by what is known as the law of large numbers, so if I throw a million darts into that square, about pi/4 times a million are going to land in the circle. So, I take that proportion, and multiply it by 4, which will give me the estimate of pi.

The screen will have a random number of seed. The number of points will be whatever number of points: how many darts I want to throw. And the screen will encompass all of these dart throws. On the right-hand side, I'm going to keep a running tally of how my estimator for pi is going as I throw more darts. As I move from the left to right, I'm going to have more darts, and you'll see that the estimator for pi is approaching 3.14159. At the beginning, every time I throw a dart, whether or not it hits the circle, it has a big effect, so it is really bouncing around here, so, these darts all hit the circle, and so, it increased the proportion closer to four. Well, closer to one, so that the estimate for pi was closer to four. Then we had a few that missed, and eventually, the estimator for pi is settling down to the correct answer. I'm going to simulate 100 points. Some of them made it, some of them just missed. But I count up the proportion, I multiply it by four, and it looks like my answer was 3.160. Now, let's run 1,000 simulations. Since I'm doing more points, it'll change the answer. It started out kind of badly, but as I did 1,000 points, the estimator for pi, was 3.140, which is all of a sudden much closer. Let's see what happens if I simulate 100,000. The answer ended up 3.137. *The answer was a little bit worse* than the run with only 1,000, but that just happens *due to random error*. *The more you run, the better your answer will be*.

### Example 3: Fun with Calculus

Now, we'll do a fun little calculus example. I'm going to use simulation to integrate sine of pi x from zero to one. The way you did it back before you learned that sine integrates to cosine, or negative cosine, actually, the way you did it before you learned how to do actual integration is that you added up a bunch of rectangles, that is how Leibniz did it in the 1600s. You approximated sine of pi x by a bunch of rectangles. You let the width of the rectangles get skinnier, and then you added them all up. So, what we're going to do is I'm going to sample n random rectangles between zero and one. They're going to have height f of x, sine of pi x, and width one over n. But instead of being right next to each other, we're going to have these rectangles centered randomly along the x-axis between zero and one, as opposed to rectangles right next to each other, and add up the areas, I'm going to make n really big, and I'm going to get the answer. The answer is two over pi.

Let's see what the demo is going to look like. I'm randomly selecting 64 rectangles. You can't see this here very easily, but there are 64 rectangles of width one over 64, and they all have the correct heights. Wherever the center is, that is their height. And I'm going to add up all the areas of those rectangles. Here is the seed, 567893. And it turns out, my estimator for the area from zero to one is 0.5886. The real answer is two over pi, which is 0.6366. So, not the best answer. But that is only because we sampled 64 rectangles. We'll see in the demo what happens when I simulate more rectangles. Now, I'm going to only simulate 4 rectangles, so you can really explicitly see what we have here. That one is centered there, and they're not very adjacent to each other. Each of them has a width one over 25. And I add up the areas, and I get, 0.5790. Not the best answer ever. Let's now do 64 rectangles. I'm not going to change the seed. Estimate, see, this looks a lot more like what we had before, and now, the answer is 0.6694. That is a little bit better, but it is in the other direction. And, finally, I'm saving the best for last. I happen to know that I can sample up to 1,024 in this software before it bags out on me, so let's do 1,024 rectangles. These are really skinny rectangles. And here is what we get. Look at that, there is 1,024 rectangles there. A little bit hard to see. And there is the answer, 0.6374, really close to 0.6366, so I'm really happy about that. And, in fact, Monte Carlo integration is a very nice methodology to use in several applications.

## Knowledge Check 1.6

Suppose there are 40 random people in a room. What is the probability that at least two of them will have the same birthday?

A. Close to 0
B. A bit less than ½
C. Almost exactly ½
D. Somewhat greater than 1/2 correct

Inscribe a circle in a unit square and toss 1000 random darts. Suppose that 800 of those darts land in the circle. Using the technology developed in this lesson, what is the resulting estimate for pi?

A. -3.2 B. 2.8 C. 3.0 D. 3.2 E. 4.0

# Module 1

## Lesson 7: More Baby Examples

### Evil Random Numbers: Box Muller Method

I'm going to show you what happens when you use a bad random generator. So, so far, I've kind of hinted that these random number generators aren't really random at all. However, there have been cases in the past people have developed random number generators that do not appear to give random numbers, and they've still been used. We'll get into more details on that when we talk about random number generation later on in the course, but I'm just going to give you an example of what you can come up with. I'm going to simulate in a simple way people's heights versus weights. What you're going to be seeing is sort of a two-dimensional normal distribution with mostly observations in the middle and some in the tails in the outside. And one dimension will correspond to the height. And the other will correspond to weight. Do the observations look random? In one case, I'm going to use a good random number generator, and they will look random. This is what is called the Box-Muller method. What we see here is this blurb in the middle here. Let's pretend the heights are on the x-axis and the weights are on the y-axis. Let's also pretend that heights and weights are not correlated, which is very unrealistic. Right over here, this is a big, tall, heavy guy and down here is a shorter very light person. But these people are in the tails obviously. See the tails are many fewer observations, and most of the action is taking place in the middle here. Just like you would expect from a normal distribution. And as before, I'm going to keep track of the number of points that I simulate as well as the seed. And I'm going to have both a good generator and a bad generator and we'll see what comes up here. I've got a slightly different graphic. But you'll have access to this software as well.

Let's generate, 1,00 points. You can see that we have this cloud of points that's generated over the x-y axis. Most of this stuff again is taking place in the middle. There's a little bit on the outside. Let's do 1,000 observations instead. and again, you can see most of the stuff is taking place in the middle, and you get some of this tail action going on. Now, I'm also going to generate these random numbers using a bad generator called the random generator, and it has probably bad performance characteristics. Actually, it is not so bad it turns out, but I think that's just a random chance. So why am I having a cow over this random varied generator? A generator must work well for any seed that you can think of. And I happen to know that this 12345, wasn't going to give me very bad performance. What happens if I pick a power of two (1024) as my seed? Here's what you get. It's not at all random. And this has to do with the fact that rando is just not a good generator. It's got some probably bad properties. And we'll talk about those later on in the course. This is just a proof of what's coming up.

### Queues `R Us: MM! Queue Simulation

Now we'll take a look at a queueing problem. Suppose we go to McWendy's, which is a popular burger joint. I encounter a single server queue and there's one line and there's one server at the front of the line, and people go in first in first out. This is the simplest possible queueing model. What happens is the arrival rate approaches the service rate? Does the line get pretty long? Do the hamburgers start to taste better? Well, let's see.

Here's what the typical demo screen looks like. First of all, I would specify an interarrival mean. And what that means is customers show up around every 4 minutes, plus or minus. The service means about after 3 minutes; a guy can get served. In other words, the server is a little bit faster than the customers showing up. The MM1 notation means this is a so-called memory list or M: exponential or Markovian interarrival times, M: exponential service times, and one server. So, you can see that this is a moderately congested system because this ratio 3/4 (0.75) is starting to get near one. If this thing got to be .999, you'd be in a major traffic jam. Anyway, let's go over and see how we would simulate this. The I in this column refers to the customer number showing up in the queue. Arrive means the arrival time. Start means when they start getting served, Service is the service time, Leave is the leave time when he complete service, and Wait is the amount of time he waits. These graphics simply represent the current queue length as a function of time. And the server utilization. It turns out in this example, the server's probably going to be utilized 75, 80% of the time, and the queue length bounces from zero to up to four or five and back down.

The first guy, he shows up at time zero. He showed up when the store opened. He didn't have to wait at all. His service time turns out was seven. If the mean of three gets plugged into the algorithm, you could get a seven. Since he starts getting served at time zero, and the service time is seven, he leaves at time seven. Meanwhile, the second guy shows up at time two. Now, on average, the time between arrivals is four, but he showed up two minutes after the first guy. The problem for this guy is that somebody's getting served already and the first person doesn't get out of service until time seven. So, customer two has to wait until time seven to leave. So, he waits five minutes. Now, finally, after he waits a little while and he gets served at time seven, his service time itself is three minutes. So, he leaves after seven plus three, he leaves at time 10. The customer three shows up at time 5. Customer two is getting served until time 10. So, he has to wait until time 10. When he finally gets to get served at time 10, his service time is five more minutes. And he leaves at time 15. You can see the respective service times for the next customers in a spreadsheet very easily. Give you the expected, actual waiting times. Overall, the first customer waited zero. Second, five, third five, the fourth, fifth and sixth customers didn't have to wait at all, good for them. The seventh customer waited one minute. Well, look at this, we had a busy period here where a lot of people had to wait. This is what the simulation tells you.

What's nice about this simulation and all simulation products is that they give you summary statistics. This output analysis summary statistic page says that after time 93, we had a throughput of 21 customers. The average number of customers in the system during this run was 2.3 customers. The average time in the system was 10.19 minutes. The server was being utilized 93% of the time. The average length of the queue was 1.37. The average time of the customer was 6.05. It gives you all the output that you would ever want from the simulation, and it's all automatic. You don't have to do the programming yourself.

## Stock Market Follies

The next example that I'll give has to do with stock market follies, and I just made these numbers up. But you can see that there's a tremendous application in the stock market. So, I'm going to simulate a small portfolio of various stocks. I'll simulate them over a five-year period. And the stocks are going to change, the prices are going to change randomly from year to year. Stock portfolio will have components that have various volatilities. Some sectors are more highly volatile than others. And what you can do, I won't do this in this particular simulation, but you can consider different mixes for the portfolio that take into account the anticipated rate of return and the different portfolio volatilities, and you could optimize over these types of portfolios and make a lot of money. That's how Markowitz got his Nobel Prize.

What I'll show you now is a very simple spreadsheet application that I just made up with six different categories of stocks: energy sector, pharmaceuticals, entertainment, insurance, banking and computer technology. Energy, apparently, it goes up by about 5% a year on average. Pharmaceuticals up 6% a year. Computer technology is just racing up 18% a year on average. The standard deviations, though, leave something to be desired. Even though energy appreciates at about 5% a year on average, unfortunately, the stand deviations are 30%. This is not something that you can take lightly. Standard deviation in computer technology is 50% here, which is crazy. Entertainment industry and insurance industry, those are a little more well behaved. Here, I'm going to start out with $5,000 in each of the components for a $30,000 portfolio. And I'm just simulating what happens from year to year. Well, the energy goes up by about 5% per year. Well, look it went way down the first year to 2701. Then it went back up, stayed the same, went back down. Went back up over the five years. So, it had to do with that volatility. Same kind of thing happened over here with computer technology. It went way down the first years and started catching up a little bit in year five. And you can see that some of the stocks did better than others. And by the end of five years, we made 3,000, $4,000, or three or four million, depending on how you count it.

Let me give a demo now, and I'll show you how the volatility actually works. Here's a live demo that we'll do. If I hit F9 basically, the simulation changes its random numbers. You can see it changes from run to run. Sometimes I lose money. That's how it is when you have these highly volatile portfolios. What you would be expected to do to make sense out of this thing is that maybe you simulate the portfolio 1,000 or 10,000 times and you get a histogram of that number right there. Depending on the volatility, rate of return, or weights, you can design a portfolio that tends to do better or is more conservative than others. That's what portfolio analysis is and simulations is a big help with that.

## Taking a Random Walk

The last thing I'll talk about in this lesson is taking a random walk. What we're going to do is I'm going to be a drunk guy much, and I'm going to take a step up or down. The amount of space that I move up or down every minute or every hour is normally distributed. So, I want to know where I am after a certain number of time units. That's called a random walk which converges into what's called Brownian motion. It turns out, Einstein and Black and Scholes won Nobel Prizes for this research.

This is what it looks like. Every time it goes up a little bit, right there. I'm a drunk person walking up and down. And does this not look like a stock price? So, this type of thing has applications in stock pricing, portfolio analysis, option pricing, absolutely wonderful topic areas.

## Module 1

Here's a summary of what we did just now. We ran some simulations on additional easy examples. All of them involve randomness. And finally, next time, I'm going to show you how to generate that randomness on a computer. And this will lead to a bunch of interesting issues that will purvey the entire course.

### Knowledge Check 1.7

1. TRUE or FALSE? All random number generators perform pretty much the same.

2. Suppose customers to a barber shop show up at times 4 and 11. Moreover, suppose that it takes the barber 12 minutes to serve customer 1 and then 14 minutes to serve customer 2. When does customer 2 leave the barber?

A. 18
B. 25
C. 30
D. 40

# Module 1

## Lesson 8: Generating Randomness

I'm going to continue the whirlwind tour with a discussion on the generation of randomness. The algorithms that generate randomness are not random at all. I've already alluded to that. But, they appear to be random. So, we'll see how that's possible.

We need random variables to run the simulation. You need interarrival times. You need service times. So, the trick that we'll learn about is that you generate uniform zero one pseudo-random numbers. Pseudo just means pretend random, not really random. Uniform zero ones are just random numbers between zero and one. These pseudo-random numbers, PRN's, are generated via deterministic algorithms. So, in fact they're not random, they just seem to be. Now these uniforms are then fed into an equation. You start with these uniforms, and I can get any other distribution I want. I can get exponential interarrival times. I can get normal heights. But I need to start out with these uniforms. Uniform zero ones. Again, they come from a deterministic algorithm. The nicest one is one called a linear congruential generator. Let's make it an integer seed, X(0). I'm going to use that to generate my next number, X(i). X(i) for i = 1, is equal to some constant A times X(i-1) and that would be 0 in this case. m is usually a really big prime number and a is some other big number. But it turns out, if you use this algorithm X(i) = X(i-1) * mod m, that often works out to give you nice random integers. To change that from a random integer to a random number, PRN between zero and one, I just start with X(i) / m, which is the biggest possible integer that I can get from this algorithm. And then that's guaranteed to give me a number between zero and one. So, we'll say that U(i) = X(i) / m.

Here's a pretend example. Let's start with X(0) = 0. And then, X(i) = 5 * X(i-1) mod 7. Then X(1) = 20 mod 7 = 6, X(2) = 2, X(3) = 3, X(4) = 1,and X(5) = 5, etc. So, U(1) = X(1)/m = 6/7, U(2) = 2/7, U(3) = 3/7, etc. You can tell the numbers don't look so random. But a real example involves the following linear congruential generator, X(i) = 16807 X(i-1) mod(231 -1). Now that number is quite a bit bigger than seven. That's about two billion. U(i) = X(i)/m will give you a number between zero and one. This generator used to be used in a number of simulation languages. It's not anymore. It's got nice properties including the fact that it has what's called a large cycle time. But there are better generators out there.

Now how do I go from these uniforms to generating other random variables. Let's start with U(i) ~ Unif(0,1) and It always amounts to applying some transformation. For instance, if I start out with that uniform, -(1/λ) ln(U(i)) ~ Exp(λ), the transformation to the U is an exponential random variable. Now that's true by what's called the inverse transform method. Now, it turns out there are other more sophisticated methods available. For instance, Box-Muller, which we've already seen. That's used for the normal distribution.

Here's a summary. We showed how to generate what appears to be randomness on a computer. And in the next lesson, we'll show how random input means random output. And that spells trouble with a capital t. What do you do about that? You can't quite use the baby stat stuff that you learned when you took that course a long, long time ago. So, what do you end up doing about it?

### Knowledge Check 1.8

1. Suppose we are using the (terrible) pseudo-random number generator [MATH] mod(8), with starting value ("seed") [MATH]. Find the second PRN, [MATH].

A. 0 B. 1/8 C. [✓ 3/8] D. 3

2. Suppose that we generate a pseudo-random number U = 0.728. Use this to generate an Exponential (lambda = 3) random variate.

a. -0.106 [✓ b. 0.106] c. -0.952 d. 0.952

# Module 1

## Lesson 9: Simulation Output Analysis

The last stop on our whirlwind simulation tour will be a discussion on Simulation Output Analysis. Here's the lesson overview. Last time I talked a little bit about how we can generate randomness so as to run the simulation. Now, unfortunately, random input to the simulation means random output, and that requires very careful analysis. What can we do about that? The bottom line is that everything they taught you in your Baby Stats class was a big fat lie. The problem is that simulation output is never, ever independent, or normal. So, we're going to need some new methods, and we'll hint at those today.

### Analyzing Randomness

Let's consider some consecutive customer waiting times in line at McWendy's. First of all, the waiting times for consecutive customers are not normally distributed. Usually, they're skewed. What happens is most people kinda wait about a certain amount of time, but there's right tail. It's certainly not symmetrically distributed, not normally distributed. Also, the waiting times are not identically distributed because patterns change during the day. There might be a long line of people at McWendy's during breakfast time, lunchtime, or dinnertime, but in those times in between the customer arrival patterns are completely different, so the waiting times might be like zero during those slow times. The worst thing though is that waiting times are not independent of each other. If you're waiting in line a long time, then the poor guy next to you is probably also waiting in a long time, so your waiting time is correlated with his. So, what I've just shown is that waiting times are not independent identically distributed (i.i.d.) normal random variables. This is a problem. You are not allowed to analyze simulation output data by the usual Baby Stats methods. So, it turns out there's going to be two cases that we can consider with respect to simulation output analysis. We have a terminating simulation where you're interested in short-term behavior and use one method called the method of independent replications. Examples of terminating simulations are like you go to a bank and you're only interested in the behavior over the course of one day. What's the average customer waiting time in a bank over the course of a day? You're probably going to end up simulating individual days many times, but it's just one day. Another example might be the average number of infected people during a pandemic. The pandemic is over, there's nothing long term about it, it only lasts for a month or so, a couple months, how many people got infected? These are terminating, short run simulations. There's also a steady-state simulation of a long run simulation. For long-term behavior, let's run an assembly line 24/7 or a Markov chain, that's run for a long time. These are the kinds of things that we care about with steady-state simulation, and, in that case, you attack the problem by something called the method of batch means or other methods.

What the method does is that it makes independent runs or replications of the simulation model, each run is conducted under identical conditions. Then you look at the sample means from each of these runs, each of these replications, and you pretend that sample means are approximately i.i.d. normal random variables. There's a lot of evidence to the truth to that assumption. And then, use classical baby statistics on that i.i.d. sample of replication means, not on the original observations, on these means that you get from each of the replications. Then you can use the classical statistical techniques on those. You still have to be careful, but it works.

### Steady-State Simulation

First, you have to deal with the initialization bias. What that is, the stuff that happens right at the beginning of simulation is not indicative of steady-state. Like if you open up the store, nobody's going to have to wait right at the beginning unless there's a giant sale going on. Nobody's going to be waiting in line. Whereas long-run behavior, people are more apt to be in line, at least on average over the long term. So, I have to deal with the bias of the simulation at the beginning of the run. Usually, what people do is they warm up the simulation before they start collecting the data, the steady-state data. If you don't do this, it can mess up your subsequent statistical analysis. It turns out there are lots of ways to deal with steady-state data. I mentioned this method of batch means. There's also other funny sounding things like overlapping batch means or spectral analysis, standardized time series, regeneration. Batch means is the one that most people use. In batch means you make one giant run versus a number of shorter independent replications, because it's steady-state. You warm up the simulation before collecting data just like I warned you on the previous slide. You chop these remaining observations, after you warm it up from the one long run, into contiguous adjacent batches. So, you batch the observations, then you take the sample mean from each batch. And using various central limit theorems, you can assume that those sample means are again i.i.d. normal and bang, off you go, classical statistics on these batch means.

# Module 1

Knowledge Check 1.9

1. TRUE or [✓ FALSE]? Simulation outputs such as consecutive customer waiting times are almost always independent and identically distributed normal random variables.

2. Let's simulate a bank that closes at 4:30 p.m. What kind of simulation approach would you take?

A. Steady-state simulation
[✓ B. Terminating simulation]
C. Arnold Schwarzenegger simulation
D. I'm from The University of Georgia. What is simulation? And what is bank?