PROFESSOR: I want to pick up exactly where I left off last time. When I was talking about various sins one can commit with statistics. And I had been talking about the sin of data enhancement, where the basic idea there is, you take a piece of data, and you read much more into it than it implies. In particular, a very common thing people do with data is they extrapolate. I'd given you a couple of examples. In the real world, it's often not desirable to say that I have a point here, and a point here, therefore the next point will surely be here. And we can just extrapolate in a straight line. We before saw some examples where I had an algorithm to generate points, and we fit a curve to it, used the curve to predict future points, and discovered it was nowhere close.

Unfortunately, we often see people do this sort of thing. One of my favorite stories is, William Ruckelshaus, who was head of the Environmental Protection Agency in the early 1970s. And he had a press conference, spoke about the increased use of cars, and the decreased amount of carpooling. He was trying to get people to carpool, since at the time carpooling was on the way down, and I now quote, "each car entering the central city, sorry, in 1960," he said, "each car entering the central city had 1.7 people in it. By 1970. this had dropped to less than 1.2. If present trends continue, by 1980, more than 1 out of every 10 cars entering the city will have no driver." Amazingly enough, the press reported this as a straight story, and talked about how we would be dramatically dropping. Of course, as it happened, it didn't occur. But it's just an example of, how much trouble you can get into by extrapolating.

The final sin I want to talk about is probably the most common, and it's called the Texas sharpshooter fallacy. Now before I get into that, are any of you here from Texas? All right, you're going to be offended. Let me think, OK, anybody here from Oklahoma? You'll like it. I'll dump on Oklahoma, it will be much better then. We'll talk about the Oklahoma sharpshooter fallacy. We won't talk about the BCS rankings, though.

So the idea here is a pretty simple one. This is a famous marksman who fires his gun randomly at the side of a barn, has a bunch of holes in it, then goes and takes a can of paint and draws bullseyes around all the places his bullets happened to hit. And people walk by the barn and say, God, he is good. So obviously, not a good thing, but amazingly easy to fall into this trap.

So here's another example. In August of 2001, a paper which people took seriously appeared in a moderately serious journal called The New Scientist. And it announced that researchers in Scotland had proven that anorexics are likely to have been born in June. I'm sure you all knew that. How did how did they prove this? Or demonstrate

this? They studied 446 women. Each of whom had been diagnosed anorexic. And they observed that about 30 percent more than average were born in June. Now, since the monthly average of births, if you divide this by 12, it's about 37, that tells us that 48 were born in June. So at first sight, this seems significant, and in fact if you run tests, and ask what's the likelihood of that many more being born in 1 month, you'll find that it's quite unlikely. In fact, you'll find the probability of this happening is only about 3 percent, of it happening just by accident.

What's wrong with the logic here? Yes?

STUDENT: They only studied diagnosed anorexics.

PROFESSOR: No, because they were only interested in the question of when are anorexics born, so it made sense to only study those. Now maybe you're right, that we could study that, in fact, more people are born in June period. That could be true. This would be one of the fallacies we looked at before, right? That there's a lurking variable which is just that people are more likely to be born in June. So that's certainly a possibility. What else? What else is the flaw? Where's the flaw in this logic? Well, what did they do? They participated in the Oklahoma sharpshooter fallacy. What they did is, they looked at 12 months, they took the months with the most births in it, which happened to be June, and calculated the probability of 3 percent. They didn't start with the hypothesis that it was June. They started with 12 months, and then they drew a bullseye around June. So the right question to ask is, what's the probability, not that June had 48 babies, but that at least one of the 12 months had 48 babies. That probability is a lot to higher than 3 percent, right? In fact, it's about 30 percent.

So what we see is, again perfectly reasonable statistical techniques, but not looking at things in the right way. And answering the wrong question. That make sense to everybody? And you can see why people can fall into this trap, right? It was a perfectly sensible, seemingly sensible argument. So the moral of this particular thing is, be very careful about looking at your data, drawing a conclusion, and then saying how probable was that to have occurred? Because again, you're probably, or maybe, drawing the bullseye around something that's already there. Now if they had taken another set of 446 anorexics, and again June was the month, then there would be some credibility in it. Because they would have started with the hypothesis, not that there existed a month, but that June was particularly likely. But then they would have to also check and make sure that June isn't just a popular month to be born, as was suggested earlier.

All right, I could go on and on with this sort of thing, it's kind of fun. But I won't. Instead I'm going to torture you with yet one more simulation. You may be tempted at this point to just zone out. Try not to. And as an added incentive for you to pay attention, I'm going to warn you that this particular simulation will appear in the final, or a variant of it. And what we'll be doing is, early next week we'll be distributing code, which we'll ask you to study, about two or three pages of code, and then on the final we'll be asking you questions about the code. Not that you

have to memorize it, we'll give you a copy of it. But you should understand it before you walk in to take the final. Because there will not be time to look at that code for the first time during the quiz, and figure out what it's doing.

OK, so let's look at it. I should also warn you that this code includes some Python concepts, at least one, that you have not yet seen. We'll see it briefly today. This is on purpose, because one of the things I hope you have learned to do this semester, is look up things you don't know, and figure out what they do. What they mean. Because we obviously can't, in any course, or even any set of courses, tell you everything you'll ever want to know in life. So intentionally, we've seeded some things in this program that will be unfamiliar, so during the time you're studying the program, get online, look it up, figure out what they do. If you have trouble, we will be having office hours, where you can go and get some help. But the TAs will expect you to have at least tried to figure it out yourself. Yeah?

STUDENT: Will the final be open note?

PROFESSOR: Final will be open book, open notes, just like the quizzes. It will be the first two hours of the allotted time, we won't go the whole 3 hours, OK? So it won't be hugely longer than the quizzes. It would be a little bit longer. And again, very much in the same style of the quizzes.

All right, let's look at this. Let's assume that you've won the lottery, and have serious money that you foolishly wish to invest in the stock market. There are two basic strategies to choose from, in investing. You can either have what's called an indexed portfolio, or a managed portfolio. Indexed portfolios, you basically say, I want to own all of the stocks that there are, and if the stock market goes up, I make money, if the stock market goes down, I lose money. I'm not going to be thinking I'm clever, and can pick winners and losers, I'm just betting on the market as a whole. They're attractive, in that a, they don't require a lot of thought. And b, they have what's called a low expense ratio, since they're easy to implement, you don't pay anyone to be brilliant to implement if for you. So they're very low fees.

A managed portfolio, you find somebody you think is really smart, and you pay them a fair amount of money, and in return they assert that they will pick winners for you, and in fact, you will outperform the stock market. And if it goes up 6 percent, well you'll go up 10 percent or more, and if it goes down, don't worry, I'm so smart your stocks won't go down. There's a lot of debate about which is the better of these two. And so now we're going to try and see if we can write a simulation that will give us some insight as to which of these might be better or worse. All right, so that's the basic problem.

Now, as we know, and by the way we're not going to write a perfect simulation here, because we're going to try and do it in 40 minutes, or 30 minutes. And it would take at least an hour do a perfect simulation of the stock market. All right. First thing we need to do is have some sort of a theory. When we did the spring, we had this

theory of Hooke's Law that told us something, and we built a simulation, or built some tools around that theory. Now we need to think about a model of the stock market. And the model we're going to use is based on what's called the Efficient Market Hypothesis. So the moral here, again, is whenever you're doing an implementation of a simulation, you do need to have some underlying theory about the model. What this model asserts is that markets are informationally efficient. That is to say, current prices reflect all publicly known information about each stock, and therefore are unbiased. That if people thought that the stock was underpriced, well people would buy more of it in the price would have risen already. If people thought the stock was overpriced, well, people would have tried to sell it, and it would have come down.

So this is a very popular theory, believed by many famous economists today, and in the past. And says, OK, that effectively means that the market is memoryless. OK, that it doesn't matter what the price of the stock was yesterday. Today, it's priced given the best-known information, and so tomorrow it's equally likely to go up or down. Relative to the whole market, right? It's well known that over periods of multiple decades, the market has a tendency to go up. And so there's an upward bias to the stock market, contrary to what you may have seen recently. But that no particular stock is more or less likely to outperform the market, because all the information is incorporated in the price. And that leads to a notion of being able to model the market, how? How would you model individual stocks if you believe this hypothesis? Somebody? What's going to happen?

STUDENT: Random walk.

PROFESSOR: Yes, exactly right. So we would model it as a random walk. In fact, there's a very famous book called A Random Walk Down Wall Street, that was one of the first to make this hypothesis. Now later, we may decide to abandon this model, but for the moment let's accept that. And let's think about how we're going to build the simulation. Whenever I think about how to build an interesting program, and I hope whenever you think about it, the first thing I think about is, what are the classes I might want to have, what are the types? And it seems pretty obvious that at least two of the things I'm going to want are stock and market. After all, I'm going to try and build a simulation of the stock market, so I might as well have the notion of a market, and probably the notion of a stock.

Which should I implement first? Well, my usual style of programming would be to implement the one that's lowest down in the hierarchy, near the bottom. I won't be able to show you what a market does unless I have stocks, but I can look at what an individual stock does without having a market. So why do I implement this first? Because it will be easier to unit test. I can build class stock, and I can test class stock, before I have a class market. So now let's look at it. Clean up the desktop a little bit. This is similar to, but not identical to, what you have in your handout. All right, so there's class stock. And I'm going to initialize it, create them, with an opening price. When a stock is first listed in the market, it comes with some price. I'm gonna keep as part of each stock, it's history of prices, which we can initialize, well, I've initialized it as empty, but that's probably the wrong thing, right? I probably should have had

it being the, starting here, right, the opening price.

Now comes an interesting part. Self dot distribution. Well, I lied to you a little bit in my description of what it meant to have the Efficient Market Hypothesis. I said that no stock is likely to outperform the market or underperform the market. But it's not quite true, because typically what they actually do that, is they say it's adjusted for risk. It's clear that some stocks are more volatile than others. If you will buy stock in an electrical utility which has a guaranteed revenue stream, because no matter how bad the economy gets, a lot of people still use electricity, you don't expect it to fluctuate a lot. If you buy stock in a high tech company, that sells things on the internet, you might expect it to fluctuate enormously. Or if you buy stock in a retailer, you might expect it to go up or down more dramatically with the economy, and so in fact there is a notion of risk, and I'm not going to do this in this simulation, but usually people have to be paid to take risk. And so it's usually the case that you can get a higher return if you're willing to take more risk. We might or might not have time to come back to that.

But more generally, the point is, that each stock actually behaves a little bit differently. There's a distribution of how it would move. So even if, on average, the stock is expected to not move at all from where it starts, some stocks will be expected to just trundle along without much change, not very volatile. And other stocks might jump up and down a lot because they're very volatile. Even if the expected value is the same, they'd move around a lot.

So how can we model this kind of thing? Well, we've already looked at the basic notion. Last time we looked at the notion, last lecture we looked at the idea of a distribution. And when we do a simulation, we're pulling the samples from some distribution. It could be normal, everything, that would be a Gaussian, where if you recall there was a mean, and a standard deviation, and most values were going to be close to the mean. Especially if there is a small standard deviation. If there's a large standard deviation it would be spread. Or it could be uniform, where every value was equally probable. We also looked at exponential.

So we're going to assign to each stock, when we create it, a distribution. Some way of visualizing, or thinking about, where we draw the price changes from. This gets us into a new linguistic concept, which we'll see down here. You don't have this particular code on your handout, you do have a code that uses the same concept. So here's my unit test procedure. And here's where I'm going to create distributions. And I'm going to look at two. A random-- a uniform, and a Gaussian. What lambda that does, it creates on the fly a function, as the program runs. That I can then pass around. So here, I'm going to look at the thing random dot uniform, for example, between minus volatility and plus volatility. So ignoring the lambda, what do we expect random dot uniform to do? It has equally likely, in the range from minus volatility to plus volatility, it will return any value in here.

But notice the previous line, where I am computing volatility. If I wanted every stock to have the same volatility, I could just do that, if you will, at the time I wrote my program. But here I want it to be determined, chosen at run

time. So first, I choose a volatility randomly, from some distribution of possible volatilities from 0 to, in this case, 0.2. Think of this as the percentage move per day. So 2/10 of a percent, would be the move here. And then I'll create this function, this distribution d 1, which will, whenever I call it, give me a random, a uniformly selected value between minus and plus volatility. Then when I create the stock, here, I can pass it in, pass in d 1. OK, it's a new concept. I don't expect you'll all immediately grab it, but you will need to understand it before the quiz comes along.

And then I could also do a Gaussian one here, with the mean of 0 and the standard deviation of volatility divided by 2. Where do these parameters come from? I made them up out of whole cloth. Later we'll talk about how 1 could think about them more intelligently. Now what do I do with that? All right, we'll see that in a minute. But people understand what the basic idea here is.

Now, I can set the price of a stock. And when I do that, I'll append it to history. I can, oh, these have got some remnants which we really don't need. I'll get rid of this which is just an uninteresting thing. And let's look at make move. Because this is the interesting thing. Make move is what we call to change the price of a stock, at the beginning or end of a day if you will. So the first thing it does, is it says, if self dot price is 0, I'm just going to return. This is not the right thing to do, by the way. Again, there are some bugs in here. You won't find these bugs in your handout, right? Code is different in the handout. But I wanted to show these to you so we could think about. What I'm more interested in here than in the result of the simulation, is the process of creating it.

So why did I put this here? Why did I say if self dot price equals 0 return? Because the first time I wrote the program, I didn't have anything like that here, and a stock could go to 0 and then recover. Or even go to negative values. Well we know stock prices are never negative. And in fact we know if the price goes to 0, it's delisted from the exchange. So I said, all right, we better make a special case of that. it turns out, that this will be a bug, and I want you to think about why it's wrong for me to put this check here. The check needs to be somewhere in the program, but this is not the right place for it. So think about why I didn't leave it here.

OK, then we'll get the old price, which we're going to try and remember, and now comes the interesting part. We're going to try and figure out how the price should change. So I'm first going to compute something called the base move. Think of this as kind of the basis from which we'll be computing the actual move. I'll draw something from the distribution, so this is interesting, I'm now calling self dot distribution, and remember this will be different for each stock. It will return me some random value from either the Gaussian or the normal distribution. With a different volatility for the stocks because that was also selected randomly, plus some market bias. Saying, well, the market on average will go up a little bit, or go down a little bit. And then I'll set the new price, if you will, self dot price, to self dot price times 1 plus the base move.

So notice what this says. If the base move is 0, then the price doesn't change. So that makes sense. Interesting question. Why do you think I said self dot price times 1 plus the base move, rather than just adding the base move to the stock, price of the stock? Again, the first time I coded this, I had an addition there instead of a multiplication. What would the ramifications of an addition there be? That would say, how much the stock changed is independent of its current price. And when I ran that it, I got weird results, because we know that a Google priced at, say, 300, is much more likely to move by 10 points in a day than a stock that's priced at $0.50. So in fact, it is the case, if you look at data, and by the way, that's the way I ended up setting a lot of these parameters and playing with it, was comparing what my simulation said to historical stock data. And indeed it is the case that the price of the stock, the move, the amount of move, tends to be proportional to the price of the stock. Expensive stocks move more.

Interestingly enough, the percentage moves are not much different between cheap stocks and expensive stocks. And that's why, I ended up using a multiplicative factor, rather than an additive factor. This is again a general lesson. As you build these kinds of simulations, or anything like this, you need to think through whether things should be multiplicative or additive. Because you get very different results, typically. Multiplicative is what you want to do if the amount of change is proportional to the current size, whether it's price or anything else, and additive if the change is independent of the current value, typically, is I think the general way to think about it.

Now, you'll see this other kind of peculiar thing. So I've now set the price, and then I've got this test here. If mo, mo stands for momentum. I'm now exploring the question of whether or not stock prices are indeed memoryless, or the stock changes. And the fancy word for that is Poisson. People often model things as Poisson processes, which is to say, processes in which past behavior has no impact on future behavior, it's memoryless. And in fact, that's what the Efficient Market Hypothesis purports to say. It says that, since all the information is in the current price, you don't have to worry about whether it went up or down yesterday, to decide what it's going to do today.

There are people who don't believe that, and instead argue that there is this notion called momentum. These are called momentum investors. And they say, what's most likely to happen today, is what happened yesterday. Or more likely. If the stock went up yesterday, it's more likely to go up today, than if it didn't go up yesterday. So I wasn't sure which religion I was willing to believe in, if either, so I added a parameter called, if you believe in momentum, then you should change the price by -- And here I just did something taking a Gaussian times the last change, and, in fact, added it in. So if it went up yesterday, it will more likely go up today, because I'm throwing in a positive number, otherwise a negative number. Notice that this is additive. Because it's dealing with yesterday's price. Change, with the change. OK, so that's why we're dealing with that.

Now, here's where I should've put in this test that I had up here. Get it out from there. Because what I want to do is, say if self dot price is less than 0.01, I'm going to set it to 0, just keep it there. That doesn't solve the problem

we had before though, right? Then I'm going to append it, and keep the last change for future use. OK, people understand what's going on here? And then show history is just going to produce a plot. We've seen that a million times before. Any questions about this? Well, I have a question? Does it make any sense? Is it going to work at all? So now let's test it.

So, I now have this unit test program called unit test stock. I originally did not make it a function, I had it in-line, and I realized that was really stupid, because I wanted to do it a lot of times. So it's got an internal procedure, internal function, local to the unit test, that runs the simulation. And it takes the set of stocks to simulate, a fig, figure number, this is going to print a bunch of graphs, and I want to say what graph it is, and whether or not I believe in big mo. It sets the mean to 0, and then for s in the stocks, it moves it, giving it the bias and the momentum, then it shows the history. And then computes the mean of, getting me the mean of all the stocks in it. We've seen this sort of thing many times before. I've then got some constants. By the way, I want to emphasize that I've named these constants to make it easier to change. Starting with 20 stocks, 100 days.

And then what I do is, I stock sub 1, stocks 1 will be the empty list, stocks 2 is the empty list. Why do you think I'm starting with bias of 0? Because, what do you think the mean should be, if I simulate various things that the bias of 0? I start $100 as the average price of the stock, what should the average price of the stock be? If my code is correct, what should the average price be, after say, 100 days, if there's no bias. Pardon? 100, exactly. Since there's no upward or downward bias. They may fluctuate wildly, but if I look at enough stocks, the average should be right around 100. I don't know what the average would be if I chose a different bias. It's a little bit complicated, so I chose the simplest bias. Important lesson, so that there would be some predictability in the results, and I would have some, if you will, smoke test for knowing whether or not I was getting, my code seemed to be working.

All right, and initially, well, maybe initially, just to be simple I'm going to start momentum equal to false. Because, again, it seems simpler have a model where there's no momentum. I'm looking for the simplest model possible for the first time I run it. And then we looked at this little loop before, for i in range number of stocks, I'm going to create two different lists of stocks, one where the moves, or distributions, are chosen from a uniform, and the other where they're Gaussian. Because I'm sort of curious as to, again, which is the right way to think about this, all right? And then, I'm going to just call it. We'll see what we get. So let's do it. Let's hope that all the changes I mad have not introduced a syntax error.

All right, well at least it did something. Let's see what it did. So the test on the left, you'll remember, was the one with test one, I believe, was the uniform distribution, and test two is the Gaussian. So, but let's, what should we do first? Well, let's do the smoke test number one: is the mean more or less what we expected? Well, it looks like it's dead on 100, which was our initial price in test two. And in test one it's a little bit above 100. But, we didn't do that many stocks, or that many days, so it's quite plausible that it's correct. But, just to be sure, not to be sure, but just

to increase my confidence, I'm going to just run it again. Well, here I'm a little bit below 100 and in two, and test one a little bit below 100 as well. You remember last time was a little bit above 100. I feel pretty good about this, and in fact I ran it a lot of times in my office. And it just bounces around, hovering around 100. Course, this is the wrong way to do it. I should really just put it in a nice test harness, where I run 100, 200, 1,000 trials, but I didn't want to bore you with that here. So we'll see that, OK, we passed the first smoke test. We seem to be where we expect to be. Well, let's try smoke test two. What else might we want to see, to see if we got things working properly? Well, I kind of ignored the notion of bias by making it 0, so let's give it a big bias here. Assuming it will let me edit it. We just gotta start it up again, it's the safest thing to do. You wouldn't think I would have, I don't have -- all right, be that way about it. Fortunately, we've been through this before. We know if we relaunch the Finder. Who says Mac OS is flawless? All right, we were down here, and I was saying, let's try a larger, introduce a bias. Again, we're trying to see if it does what we think it might do. So what do you think it should do with a bias? Where should the mean be now? Still around 100? Or higher, right? Because we've now put in a bias suggesting that it should go up. Oops. It wouldn't have hurt it.

All right. So let's run it. Sure enough, for one, we see, test two, it's a little bit over 100, and for test one it's way over 100. Well, let's make sure it's not a fluke. Try it again. So, sure enough, changing the bias changed the price, and even changed it in the right direction. So we can feel pretty comfortable that it's doing something good with that. We could also feel pretty comfortable that that's probably way too high a bias, right? We would not expect that the mean should be over 160, or in one case, 150, after only 100 days trading, right? Things don't typically go up 50% in 100 days. They go down 50%, but -- All right, so that's good.

Oh, let's look at something else now. Let's go back to where, a simpler bias here. We'll run it again. And think about, what's the difference between the Gaussian and the normal? Can we deduce anything about those? Not, well, let me ask you. What do you think, yes or no? Anybody see anything interesting here? Yeah?

STUDENT: The variance of the Gaussian seems to be less than the variance of the uniform.

PROFESSOR: The variance of the Gaussian --

STUDENT: -- is less.

PROFESSOR: So all right, that appears to be the case here. But let's run it again, as we've done with all the other tests. So we have a hypothesis. Let's not fall victim to the Oklahoma sharpshooter. We'll test our hypothesis, or at least examine it again, see if it's, in some sense, repeatable. Well, now what do we see? Doesn't seem to be true this time, right? Not obviously. So, we're not sure about this. So this is something that we would need to investigate further. And we would need, to have to look at it, and it's going to be very tricky, by the way, as to what the right answer is. But if you think about it, it would not be surprising if the Gaussians, at least, gave us some

surprising, more extreme, results, than the uniform. Because the uniform, as we've set it up here, is bounded. The minimum and the maximum is bounded. With the Gaussian, there's a tail. And you might every once in a while get this, at least as we've done it in this case, this large move out at the end. You might not. There's nothing profound about this, other than the understanding that the details of how you set these things up can matter a lot.

Well, the final thing I want to look at is momentum. So let's go back, and let's set mo to true here. Well, doesn't want us to set mo to true here. Ah, there it does. So, and now let's run it and see what happens. What do you think should happen? Anybody?

STUDENT: [INAUDIBLE]

PROFESSOR: I think you're right. These ones should curl, see if I can -- oh, not bad. Let's run it. Well, it's a little hard to see, but things tend to take off. Because once things started moving, it tends to move in that direction. All right. How do we go about choosing these parameters? How do we go about deciding what to do? Well, we play with it, the way I've been playing with it, and compare the results to some set of real data. And then we try and get our simulation to match the past, and hope that that will help it predict the future. We're not enough time to go through all the, to do that a lot. I will be posting code that you can play with, and I suggest you go through exactly this kind of exercise. Because this is really the way that people do develop simulations. They don't, out of whole cloth, get it right the first time. They build them, they do what if games, they play with them, and then they try and adjust them to get them right. The nice thing here is, you can decide whether you believe momentum and see what it would mean, or not mean, etc. All right, one more lecture. See you guys next week.