ROSS COLLINS: Welcome to the tutorial on the method of simulated moments. Collaborating together on this tutorial is Armin Ashoury, Ross Collins, and Ali Kamil. We will present the tutorial in two parts. The first part, presented by me, Ross Collins, is an overview of the MSM, why it's useful, and how it works. The second part, presented by Armin Ashoury, will go through the details of how to implement the MSM in Vensim DSS.

Background information on the MSM is included in the associated book chapter with this video. The chapter goes into much more mathematical detail in the MSM formulation, which the interested viewer should consult if looking for more information.

In this video, we do not dissect every individual formula, but instead try to give a useful overview of the MSM, and then apply it to a simple example. The step-by-step replication the tutorial will likely require reading and understanding the book chapter in more detail. Still, this video should give a flavor of why MSM is useful, how it works in the general level, and what sort of time and effort is required to implement it in system dynamics models.

So what does the MSM let you do? Well, according to this quote, it allows one to "compare time series data against the same variables in a model, and minimize the weighted sum of a function of the error term by changing the uncertain parameters until best fitting estimates are found through optimization,"

This is important for model validation. So specifically what MSM does is it compares the moments of real data against the moments of simulated data or just the model output from a simulation. So moments are things like mean, variance, skewness.

Kurtosis is another one. These are the first, second, third, and fourth moments respectively.

An important thing to realize is that unknown parameters in a model will alter the simulated moments. So if you have some unknown parameters, x, and some moment, y, if you change x, then y will change.

So what the optimization method does-- referred to in the quote up at the top-- is it will find values for our unknown parameters that minimize the weighted difference-- and I'll get to why weighted is important later-- the weighted difference between real and simulated moments. And these weighted differences are called the error terms.

So we're going to apply this MSM to a simple obesity model. We basically have weight data in Excel-- this Excel file is included with this video-- we have weight data for a population of 1,000 individuals at five different points in time, specifically years 1, 5, 10, 15, and 20. The moment conditions we use are mean and standard deviation. And given that we have five years of data and two moments, we have 10 moments in total.

So the unknown parameters that we're trying to estimate through the MSM technique are called Overfeeding, Starvation, and Energy Intake Extra Trend. So I am going to show this model really quick. It's a very simple model that tracks the weight gain of 1,000 individuals. You can see you have the stock structure here, an array of 1,000 people, so we're utilizing subscripts to denote these 1,000 individuals.

And the major feedback is that as weight change accrues, you have some energy intake balance that changes. This feeds back into one's energy intake through the starvation and overfeed parameters and then alters the weight change.

So the three uncertain parameters are here. We have an underlying trend in energy intake that impacts the weight trajectory of each individual. And then we have the starvation and overfeed parameters that impact this major feedback. If energy intake balance is on the heavier side, starvation kicks in with some parameter of force. From the lower side, the overfeed parameter kicks in with some amount of force.

So let's see. If we click on the weight here and show a graph, we're only going to get to show 16 of the individuals, but you can see the trajectory of them through time-- over the 20 years. The initial conditions are captured here.

The mean of individuals is 80 kilograms-- so you can see it sort of circled around there-- and the standard deviation is 5 kilograms, which is why the initial conditions differ. But you can see the trajectories, and we have this for 1,000 individuals. So then the idea of the MSM is you have these 1,000 trajectories, and you have the moments that define these trajectories. And we compare that against the real data that we have in the exercise.

So let's see. Final slide that I'm going to talk about before I hand it off to Armin to go into the Vensim implementation. So basically, there are four overarching steps and four Vensim models associated to each step. So in the first step, we are basically making the first cut at estimating these three unknown parameters.

And to do that, we need-- and I'm going to pull up the book chapter here-- we need to minimize the weighted differences of these parameters. And that's what you can see here in equation three in the book chapter. This theta hat is the set of parameters-- in our case just three-- and we want to minimize the difference between simulated moments and the real moments to use for data.

And we have here in the middle this W, which is a weighting matrix. And the purpose of the weighting matrix is to give less weight to really noisy and uncertain moments, since those are going to sort of hijacked the optimization process when trying to choose parameters.

So the first W that we use in the first step is essentially just taking the real data moments and squaring them and then taking the reciprocal. And those are along the diagonal elements of W. Armin will go into more detail about that, but that's our initial starting point for W.

So the output of that are an initial set of parameter estimates, not necessarily optimized, but certainly informed by this initial weighting matrix. The second step is to calculate essentially a more efficiently weighted W, or what we call W*, and it uses an algorithm that is discussed in the chapter. It's equation four down here.

As you can see, it's reasonably complicated and requires first calculating the variance-covariance matrix of the simulated moments, which again we can do in Vensim-- even though it's easier in MATLAB-- and W* is simply the inverse of this matrix.

So once we've got W*, we basically go into the second step, which is very similar to the first step. It's an optimization yet again, but now we have a more efficiently weighted W. And that gives us a better set of optimized parameters.

The important thing to note is that we can iterate if we are not happy with W, but the final step involves calculating competence intervals for each of our uncertain parameters. Again, in our case, we have three of them. So these confidence intervals are just like other confidence intervals and statistical exercises. We want those bounds to be as tight as possible, and we don't want them to include zero.

So that is the overview of the MSM. I'm going to hand it off now to Armin, who is going to go through the major steps actually implemented in Vensim. So thank you for listening.

ARMIN ASHOURY: In each model, we have a part that captured the moments we need for the MSM process. So here, for example, in the FirstStep model, we see that we have the main model of obesity. And here we have the part that tries to capture the different moments that we have. For example, here we want to catch the mean of body weights and also similar deviation of body weights.

Simply, this part tries to create a matrix of moments. For each column, we have one simulation. So here, for example, we have like 10 simulations. And for each row, we have one of our moments here. Since we have like five years of data, which are year 1, 5, 10, 15, and 20, we have to use two moments, which are mean and standard deviation.

We have like 10 rows that, for the first five years, we have first five rows, we have the mean of our years. And for the second five rows, we have a standard deviation of our years. So we can create this matrix in whatever way you want, but here we try to capture these moments and create that matrix using second flow system.

In the FirstStep of moment, we tried to find three unknown parameters of obesity model, which are extra trends, starvation, overfeed. We use formula five in the book chapter and try to optimize those parameters based on that formula.

So here in formula five, you see like different parameters we have. For example, M of s is the average simulated vector of moments. So for example, in our example, we have like 10 simulation for each one of those moments, so we average the moments over those simulation and create a vector, which is M of s here.

And M of d is the real moments values in that vector that we capture from the real data we have. W here is the initial value for the weights. For the initial value, we use the Formula One over moments, real moments, to power of two. In the model, we call the function-- a function here as the error term.

So here in the view 2 of our FirstStep model, we can see that we have our three unknown parameters here. We put them all together in one vector here called parameter so that we can deal with them easily. So from the obesity model, we get the moments that we need here.

So if you take a look at the moments here-- the moments we have here-- in the last year-- year 20-- you'll see that we have matrices that has 10 moments in it, and for each moments we have like 10 simulations. So we have 10 in 10 matrix for each moments in each simulation. So we get those values.

And here we can get an average over those 10 simulations, and we create the M of s that we talked about here. So using that and using the real data that we have we calculate the difference between the moment-- the selected moments and the real moments.

And here, we see that we calculate the W based on the formula we have for the initial value, which is one over real moments to the power of 2. And using these data, we calculate the error terms here, which we just calculated for the last time of the simulation, which is the year 20.

So as you can see here, this formula here is a formula 5 from the chapter books that we have. And finally, we have to do some optimization on the model to get the value. So here, you'll see that we want to optimize over the error terms, and we want to minimize it. So we put error term here on base minus one. You have to do to choose a policy for this optimization.

So we are what we want to optimize over the parameters, so we have to choose the parameters as the variable we want to optimize. And then we do the optimization and you will get the values we need. So we do optimization, and it may take for awhile, maybe like 20 or 30 minutes.

So we are fast forwarding to the values we have at the end of that. So as you can see here, we have optimized parameters-- optimized values for our parameters, which are kind of close to the real values that we have. So in the next model, which is the CalcW model, we try to calculate the weight we need for the MSM process using other models.

So here we import optimized, unknown parameters from the previous model, which is the FirstStep model, to CalcW model. Then we calculate a variable called S using the formula for in the book chapter that you can see it over here. And the inverse of that S variable is the weight that we need for our MSM process.

In the first view of CalcW, although everything is the same as this previous model, in the second view, we try to calculate the S based on the formula we have in the book chapter. So here we can see that we try to calculate the average of simulated moments. And this is exactly the same as the previous model, and we try to calculate the average of each moment over 10 simulations.

And after that, you can see that that average of simulated moments here in the formula 4. And we have to subtract the real value of moments from that average and then multiply to its transpose and get another average over that. So here in the moments differences variable, we try to calculate the formula we have in the book chapter formula 4. And using that, we just calculated the S hat for the last final time of simulation.

We only need to calculate the value of S just for one time, so that by inversing the value of the matrix S, we can get the value of matrix W that we need the MSM process. So we just want the models for one time. And here, after the simulation ends, you can see the value of S hat we need in the final time of simulation here.

So we have like 10 by 10 matrix that has like difference moments, and we have to inverse this matrix to get the real weight matrix that we need. In the next model, which is the SecondStep model, we again want to find all known parameters that we have using some optimization this time using the calculated w from the previous part in the CalcW part.

So we imported S matrix from the previous part. And by inversing that S matrix, we get the W that we need. And also, we can import optimized parameters from the FirstStep model, or we can just use some random other variables, values as the initial values of the parameters.

And so finally, we do another optimization to find our three unknown parameters. And we again use the formula (5). Just this time, we have the W calculated from the previous model, which was the CalcW model. So here in the SecondStep model, in view one, you can see everything's the same as the two previous models.

And in view two, you can see the only [? serious ?] thing, that difference from the FirstStep model, is that we import the value of S and then inverse it to calculate the W instead of calculating W based on the data moments that we have. Here we have some kind of [? bargain ?] Vensim.

And if you try to import the values of S hat from previous model using some kind of function, like GET VDF CONSTANT or any other function that tries to import the values from the previous model, then my Vensim cannot calculate the inverse of that metrics [INAUDIBLE].

So here we just copy-pasted the values of the calculated S matrix from previous model and just put it here as the exogenous variable, and here we calculate the inverse of my matrix, instead of just using the imported values from the previous model. For the SecondStep model, again, we have to optimize the parameter.

So again, we have like the same system as the FirstStep model. We tried to minimize the errors terms, and we are trying to optimize the parameters. And after like the simulation-- simulation may take like an hour or so-- so after the simulation ends, we have optimized value for our unknown parameters.

After the optimization is finished, we can see the optimized parameters value here. So we get these values for each one of these parameters. And we can see, for example, 0.002 is for [INAUDIBLE] random, 0.35 is for starvation, and 0.13 is for an overfeed parameters here,

The next model, which is the Confidence Interval Model, we try to calculate the confidence interval of our unknown parameters. So here we need to import the optimized parameters from the SecondStep model and the W from the CalcW model. And then we calculate the confidence intervals first by calculating a variable called Q. We can see the formula for calculating the Q here.

And then using that Q, we calculate the confidence interval using this formula we can see here. We just have to add and increase our values from each parameter. So here the confidence level factor, for example for 95%, the confidence interval-- assuming we have normal distribution-- is 1.96. Then we have to multiply that value, that confidence level factor to a square root of q, and adding and subtracting from the parameters. So after that, we have the confidence interval for each one of our parameters.

So here in the Confidence Interval Model, we can see that again we calculate the average simulated moment, and we get the S from the previous part. Also, we have two import the value of parameters meters from the previous model, which was the SecondStep model. So we get the optimized parameters here. We get the value of S here. And here we try to calculate the confidence interval using the formula that we have.

So here in the formula 7, you can see that we have 1 plus 1/K. K here is number of simulation. So for example, we have 10 simulation. Then we have delta as minus 1, which is the W. And again, delta-- delta here is the change in moments of a parameter by changing the parameter for epsilon or [? theta ?] [INAUDIBLE] value.

So for example, if you add 0.001 to an optimized value of "Starvation", how much it's average simulated mean may change or standard deviation may change to two moments that we have. So we tried to calculate that delta here using some [INAUDIBLE] variable. In [INAUDIBLE] variable, we just add and decrease a value of epsilon to each parameters, and then we get the delta here. Using this system that you can see here, we calculate the confidence interval we need.

So again, we need to run the model only for one time. Just by running the model just for one time, we can calculate the confidence interval that we need. So after running the model, we can see this we have the confidence intervals for, for example, upper bound confidence interval for each one of the parameters here, and the lower bound of confidence interval for each one of the parameters that we have in here.

The base obesity model and the moments, the capturing moments structure that we have in all four models, are kind of the same, except for the last model, which is the confidence interval model. We have to add one other set of scripts that is for sensitivity analysis. For each one of the parameters, we have one upper bound and one lower bound, so we have to add six subscripts to the model.

And for each one of the subscript, we have to run a simulation to, for example, catch the upper bound of one of the parameters or catch the lower bound of the parameters. So you have to change the model in a way that it runs the simulation for each one of those subscription. But everything else is the same in our four models for the base of the obesity model and the moment capturing model for a structure.

Finally, if we put aside the [INAUDIBLE] in Vensim and assume that we can just import and export [INAUDIBLE] between models and calculate the inverse of matrix without any problem, we can create just one common script that runs our sequential models and just import and export the data between them. And after running all the models and processes, and it just gives us the final solution of the optimized parameters and their confidence intervals.

So here for example, we created a common script that tells Vensim to optimize the first model, then import the value from the first model and calculate the W while you're running the second model. And after that, just import the W from the CalcW model and just do another optimization to get the second to do another optimization on SecondStep model.

Finally, export the values to the confidence values of unknown parameters to the confidence interval, and just calculate the confidence interval. So using such as system, using such a common script system, we can just do one-- just gives Vensim one common, and it will just do the whole process of our series models one after each other, and do the whole thing together. And just gives us the final parameter's value and their confidence interval. So that was it. Thank you so much for watching.