PROFESSOR: All right. So we're going to now show some of the examples. And so again, they're in our directory here, Examples. We're now in section 2, so we're going to go to Applications. And we have this section called Perfect Power Law. The actual example codes are all these ones called PPL 1, 2, 3, and 4. We have actually a lot of other functions in here, which are actually really useful functions. They're the functions that do all this stuff. And maybe they should actually get folded into the main D4M distribution. They're here right now. They probably could be cleaned up a little bit as part of the homework. But these are, again, very useful functions for allowing you to generate and fit these types of things.

So I have already started my MatLab session. There we are. So we will do the first one. Going along here. Figures. So if we go and look at what we did here-- so the first thing we did is we set our parameters for our Perfect Power Law fit. So we set alpha at 1.3, a D max of 1,000, and then approximately 30 bins. We called our function here, which is in this directory, which is Power Law Distribution, which will create a power law distribution from these three parameters. And then the first thing we did is just plot a distribution. So we go here, then we go to Figure 1. You can see there is that distribution. So our Perfect Power Law distribution with these parameters, very nicely done.

Then moving along, we can compute. This is how you compute the total number of vertices from the distribution. We just sum. It could be the total number of edges. We can sum there. And then we see that the total vertices was 18,187, 84,000 edges with an edge to vertex ratio of 4.6. We now have a function called edges from distribution. So if I pass in the distribution-- essentially the degrees and the counts-- it will then generate a set of vertices that is consistent with that distribution.

And now I'm going to create some permutations, basically, of these vertices. I'm going to create a-- randomly permute the edge order with these two sets here. So if I make a random permutation of the number of edges, that allows me to permute the edges. And I can also randomly permute the vertex labels themselves, and then I can look at these different

permutations to see what kind of grass they've created.

So if I don't permute the data, I just have a0 here. I just pass in the vertices that were created. I create a non-permuted adjacency matrix. I can then look at that. And that just shows you-- this is the adjacency matrix. This is in the plot. It just shows you that I just created a vertex that's entirely self loops. Not very interesting, but nevertheless, completely consistent with the perfect power of law degree distribution.

So likewise, if I just permute the edges-- so basically, I take my permutation here, and I just permute that-- this is just not permuting the vertex labels, but just permuting the edges. To the next figure here, 3, and we get this distribution-- basically start vertex, end vertex. Looks fairly random. However, you see the highest degree vertices are up here at 0, 0, because that's the order in which the generator spits them out. It gives you the highest degree vertices first. And so although we've reconnected them with different edges, you see that their vertex labels still have an intrinsic order which is corresponding to vertex degree.

I should say this generates this naturally. It's a very common way to-- if you have arbitrary vertices and you want to put them on an adjacency matrix, one of the first things to do is to reorder them according to degree. You'll always get some kind of structure that emerges there, and it's often an easier way to see what's going on there. And as you can see from the previous data, you saw if you ordered things by degree and you saw this type of [INAUDIBLE] aha, that looks a lot like a power law distribution just from the scatter of the dots.

So moving on here. So now we're just going to permute the vertex labels. So we are not permuting the edges, just the vertex labels. So if you see here-- and then by permuting the vertex labels, it looks random, but fairly sparse. That's because we still-- every single time, we've moved the vertex labels around. But all the edges associated with that vertex pair still will ship around [INAUDIBLE]. We haven't done anything to break them up. Whereas before, we broke up the edges, but we didn't change the labels.

And so then the final one, which is the one that we typically do when we want to create random, is we permute both the vertices and the edges to create a truly random looking adjacency matrix. And there you see you have something. And now what you can also see here is these high degree vertices do stand out. This is a very standard power law distribution. You'll see these dense rows and columns in there that are indicative of these very high degree nodes.

When we just permuted the edges but not the vertices, these high degree rows and columns all got shifted up into the corner of the plot. So again, another way to begin to look at the data in a way to recognize structure. And I think this adjacency matrices, once you start looking at them, you begin to get a comfort zone, just as we do with other types of data. We begin to learn what they look like. This is a very good way to look at the data, because you can look at a fair amount. If I were to actually plot the graph of 18,000 vertices as a traditional graph of the vertices and lines connecting them, it would just be blue. The entire screen would be blue. There would be no way to properly position those vertices.

Here, this is 84,000 data points, and I can still kind of see it a little bit. This is the limit. 100,000 edges is the limit of what you can put on any plot to really see it, unless there's some hidden structure that allows you to really, really move it all together. But the adjacency matrix is a great way to look at fairly large graphs. So we generally do that.

So this is the procedure. Create your perfect parallel distribution, create some edges, create some permutations, and then permute it, and then you off you are. You've just created a randomized perfect power law graph. Again, this is an example of a code where you probably might even just take this program and adjust it to suit your needs. And then, I think, we did that. And then finally, we plotted the degree distribution of a 3 just to show you that I'm not lying. And again, the triangle is the original and the blue was the thing. And you see throughout all those permutations, our degree distribution remained the same. Even though they look-- those would be completely different graphs, their degree distributions are identical.

So let's move on here. So this is just showing-- so we actually rolled this up all into one function here for you. Ran power law matrix, if you give it an alpha, a D max, and an ND, it will do those three steps for you that I had in the previous chart all in one thing and produce an adjacency matrix that's a perfect power law based on these parameters. And again, you saw we have the same number of vertices that we saw and edges and ratio that we saw before. So that's all the same. Now we're going to transform this data, clean up this data by making it unweighted, undirected. We're going to eliminate self loops. We're going take the upper triangular part. Here's another one that's unweighted, no self loops, different versions of them. And then we do-- you can see what those look like.

So if we look at the first one here, this just shows the unweighted, what basically making the data unweighted does to the data set. So the triangles are the original data, and just making it unweighted, how it distorts that data set. This shows you what happens when you make it

undirected. So unweighted means we took any cases where we had vertices with more than one connection to them, if something had five connections, now it just gets one connection. And so that was a fairly big distortion.

A perfect example is if you take a person's social network graph, and if you were to make it unweighted, what you're saying is that the connection you have with your spouse is identical with someone that you emailed once or that you friended once. And I think we all agree that that's a fair amount of information that's lost there. And so again, encouraging folks to be aware of that and to be careful of when they're doing it. Again, making it undirected just means we basically-- if I phone you a lot or I cite you a lot in papers,

That's the same as you citing me a lot, which you lose some information there. Again, this shows the kind of distortion that we get from making it undirected. And again, this shows what happens when you do no self loops. Well, there's not a lot of self loops, so we only have affected a few vertices here. So in this case, eliminating self loops is not a terribly distorted-- doesn't really distort the data very much at all.

And then finally, this shows the upper correlation matrix. So when we correlated the two, basically multiplied the adjacency matrix together, again showing what we saw before. Moving on. You can see the plots that got eliminated from my PowerPoint. MATLAB has defeated PowerPoint's attempts to deny you your education. So again, what we're doing here is we're creating a perfect power law. This is a bigger one. I wan a lot of vertices. So this time, we had 50,000 vertices, 329,000 edges with a ratio of 6 and 1/2. We create our vertices. We randomize the edge order, et cetera.

Now we're going to randomly pick a subsample of these. And what is F samp set at? It's 1/40. So I'm going to take 1/40 of all the vertices. Now I'm going to go and compute that degree, and I'm going to basically subsample all of these. And later, what you'll see-- so let's just take a look at that. So this just shows that chart here. So this is the original data. Again, this is the vertex. We're sorting the vertices by degree here. So this is the highest degree vertex. These are the lowest degree vertex. And each vertex is getting a dot here. So we have 50,000 vertices. They all get a dot here, and we've only taken 1/40 of them. And this just shows you here-- If you only take 1/40 of them and then compute their sample, this is what you get.

Now, standard sampling theory would say aha, well, I know how to correct for this. The way I correct for this is I just multiply my sample data by 40. And we took 140, so that means

whenever I measure it, the true value should be 40 times higher. So we can look at that in figure 2. So this is the true sample.

So again, we see for our high-degree vertices here-- this is the highest degree vertex here again. So this is the high-degree vertices. These are the low-degree vertices. I don't know if I said that, opposite when I mentioned it before. This is the highest degree vertex. And you see that by sampling the data, we're doing a very good job on the high-degree vertices. We're sampling them just fine. And that's why statistics works. If something is really not rare and you sample, you're going to get a good estimate.

However, for these low-degree vertices over here, what you see-- by multiplying by 40, we're significantly over-estimating their probability. As I say, this is the curve that proves that optimists and pessimists are both correct. There are so many rare things. If the world is a power law distribution, it means that there are so many rare events in the world, some of them are going to happen to you. So it means if you're an optimist, go play the lottery. If you're a pessimist, it means that lightning could hit you, and you better just stay inside. So there's just so many rare things that some really rare things are going to happen to you in your lifetime. Most likely, those rare things are very mundane.

But we can correct for this. And so we have basically a way here of deriving calculations. So we compute the parameters of our distribution, and through these two functions, compute degree correction, and apply degree correction. We can actually go back and say all right, given that we believe the data is power law and we've sampled it, can we then come up with a more uniform correction that basically gives us a better estimate that works at both the high and the low end? And that's what you see here. So basically, we haven't changed. The correction hasn't changed here. But we've downgraded these lower ones.

And essentially, what we're doing is instead of just using the average as the statistic, we're using the median. So we're using, essentially, a quantile-based correction here, a 50th percent quantile-based based correction here. And that causes us to lower the estimates of these vertices. And so it would be a better estimate and allow you to do the sampling of that. Very good.

And our final demo. So now what we're doing is power law fitting. And so we have the routines for doing that. So again, here's our distribution. It's a power law of 1.3. We've set D max to be 2,000 and about 60-ish bins. We create our parallel distribution. It has 50,000 vertices,

329,000 edges. Ratio is the same as the one we did before. We're going to make it undirected-- undirected and unweighted, undirected, unweighted, no self loops, so standard corrections that we do.

We're going to compute the degree distribution of that data and plot it. Or actually, get it there now. I'm going to then-- we have this function called power law fit. So if I compute-- so I compute the degree distribution. So we have this function called out degree which gives us the distribution. And I can find, essentially, the number of values with the one, and the one that-- our maximum, so this is estimating our poor man's slope. So we're computing the slope. We're counting the total number of edges here, and then we have this function called power law fit. Basically, we can plug in what the estimated alpha is, what the number of vertices is, and the number of edges is to find our best fit distribution.

So this basically inverts those formulas I showed you, which is given a degree distribution that sums to a particular number of vertices and sums to a particular number of edges, can you give me a new D max and a new ND, these parameters that don't really have as much meaning, to do that? And so basically, we use, essentially, a combination of three different techniques here. Because this is so nonlinear, and there's this-- basically, remember I talked about integer bins and logarithmic bins?

Well, if you look at that plot, it showed there's that bending. It's a very nasty manifold, the surface of this function. And it has a continuous part and a discrete part. So what we do here is we do essentially a sampled search where we randomly sample, looking for a location. We do a heuristic search, which is a simulated [INAUDIBLE] search. And we also use Broyden's nonlinear-- essentially a variation of Newton's method to all try and find the best set of parameters that will fit this data.

We rarely get an exact match. But you can see here it's choosing different ones. And this gives you how it's doing. From the sample search, this shows you the number of vertices and the number of edges it was able to achieve. The heuristic search didn't do very well at all, and then the Broyden search did a pretty good job, and it got us pretty well. So actually, it ended up comparing all of these, and it ended up choosing the sample search-- this one, this first one I did. It liked that best of all.

So we'll look at that here. So Figure 2 was the original data. Figure 1 shows you the manifold of this space. And so plotted in this coordinate system of n versus m, the dot is the dot that it

found. It's actually here in this [INAUDIBLE] these lines show the boundaries. But you can see this very nonlinear manifold here. This is the continuous regime. This is the integer regime. It cusps right at the transition. Again, a very nasty function to try and invert, which is why we used all those different techniques to invert it.

And then Figure 3 shows the results. So this black line shows the original model input that we provided. So that was the true model. The circle shows the data after it was transformed. We made it undirected, unweighted with no self loops. That's this. Alpha is-- this is our poor man's alpha. What you can see is almost identical to the original model. So the poor man's alpha does a very good job of fitting in this case. The triangle shows the model fit. So when we fit the data, we came up with that best fit it shows, and we then created a new distribution. This is what it looked like. And then the plus sign shows us rebinning that data onto the bins from the model. And you see that we've done a very nice job of recovering the original power law distribution even after we did that distortion.

So again, that is the last example. So I want thank you. And then for the homework, a lot of you did Homework 2, which is great. I think Homework 3 wasn't such a great hit. I'm going to definitely rethink that one. But the next homework does not require you to have done homework 3. If you did homework 2, basically, it's just saying compute a degree distribution.

And in fact, you don't even need to have done Homework 2. You can compute a degree distribution on your Homework 2, or you can compute a degree distribution on any data set. For example, today is Halloween. If you go trick or treating with somebody-- yourself, your children, somebody else's children, you can maybe email me the histogram of your candy and plot it on a degree distribution, and maybe compute the poor man's alpha coefficient from that. And the other coefficients from that would be a fun exercise to do. So that'll be the next homework. I'll email that out in the next couple of days.

So look forward. Again, if you send me the homework prior to next-- actually, just a reminder, no class next week. This room has been taken. But again, if you email me the homework prior to this time next week, I will give you feedback on it. You can still send me the homework after that. I just won't give you any feedback on it. So thank you again, and look forward to seeing you in two weeks.