[SQUEAKING]

[RUSTLING]

[CLICKING]

VIJAY GADEPALLY: I hope you all are enjoying the class so far. It's always interesting-- we have people with a variety of backgrounds here, so it's just great to kind of hear the questions as we're going along.

So I'm going to give a quick example to begin the class right now that kind of drive in some of the concepts that we talked about on the first class. But this is kind of real rather than sort of cartoon neural networks. But it's sort of a real research result that we have.

Before I begin, I'd like to thank Emily Do who was one of my graduate students who actually did all the work. I just put some of the slides together-- not even all of them, just a few. So really, the credit for this work goes to Emily. Anything I misrepresent is my fault, not hers. She graduated, so I'll take the blame for anything that's not interesting. So with that, we'll begin.

All right, so the overall goal of this project was really to detect and classify network attacks from real internet traffic. And in order to do this, as many of you can imagine, we had to find a data set that was of interest. So this is probably a problem many of you are currently thinking about, which is what data set should I get my hands on? Right, so we have a variety of data sets. Some are sensitive in nature, right? So think of internal network traffic that we're trying to collect. No one's going to let us hand this over to graduate students to kind of work on, and then more importantly, publish.

So the first thing that we wanted to do was look for a data set that was kind of open and out there. And we're fortunate that there is a group in Japan called the MAWI working group, which stands for the measurement-- and I'll use my cool tool-- the measurement and analysis of wide area internet traffic working group, that actually has 10 gig link that they've tapped using a network tap. And they actually make this

data available. It's actually continuously updated even to today.

So that's really cool, and this is within that. It's called the Day-in-the-Life of the Internet. And so this has been going on for multiple years. The data set is reasonably large when you convert it into what we call an analyst friendly form, which is something that you or I could read and make some sense out of. It's about 20 terabytes in size. So a reasonably large data set, not going to fit on a single computer or a single node, so certainly a good use case for using high performance computing or supercomputing.

And one additional piece that we should note-- and this is something that, as you're starting off your projects, you may also think about-- IP addresses are often seen as reasonably sensitive. So you can see where traffic is coming and going from. So in this particular data set, they've actually deterministically anonymized each of the IP addresses. And as you're downloading the data, you have to, essentially, sign a user agreement saying that you will not attempt to kind of go back and figure out what the original IP addresses were.

So as you're coming up with the data sets-- and I think Jeremy is going to talk a lot more about this in the next part-- you might think about, you know, are there certain fields in the data that I'm collecting that might be deemed sensitive, either today or in the future? And if so, are there just simple techniques that I could use to anonymize the data? And then with a decent end user agreement, you might be able to enforce some level of people not trying to break it and trying to go back. It's not impossible to do it, but any legitimate researcher who is using data really shouldn't care about what the original IP addresses in this particular case are or maybe other such data within whatever your collecting.

All right, so just a quick definition of anomaly detection. I really love this definition of what an outlier is from a paper by Hawkins. So, "An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism." So when we're looking at network traffic, that's what we're looking for. Right?

We're looking for these mechanisms that are present in network traffic that deviate so much from normal behavior that it arouses suspicion. And it could be for a

variety of reasons. It could be somebody uploaded this cool video of somebody falling flat on their face, or it could be a botnet, or DDoS attack, or something like that.

So outlier can be sometimes referred to as an anomaly, surprise, or exception. And within the context of cyber networks, these mechanisms can be botnets, C&C, or command and conquer servers, insider threats, or any other network attacks, such as distributed denial of service or port scan attacks.

There are a number of general techniques to deal with outlier detection. So the first one is to look for changes-- is to essentially use statistics. You look at what's going on in your mechanism, and you look for statistical anomalies from that. And you'll highlight those anomalies, and then you'll kind of drive deep into that. You could do clustering. So if you're looking at unsupervised learning, you could cluster your data. And things that are really far away from existing clusters, or known clusters, are probably something that you want to take a look at.

You could lose-- Similar to that is also distance-based techniques. So these are kind of closely related that you look for observations that very significantly from other observations in some sort of a feature space. And finally, you could use a model-based technique, where you attempt to model the behavior of normal-- right, and I use air quotes for the word normal-- and then, you come, and you look for things that deviate from this background model. So all of these four techniques are approaches to anomaly detection kind of related with each other. It's not a hard and fast rule about how they deviate from each other.

But for the popularity and the complexity of network traffic, we decide to go with a model-based technique because we found that any other means of trying to represent the data would be difficult. But this is sort of a top down approach of how we kind of thought about, well, let's look at network traffic-- let's think about anomalies in network traffic-- and let's come up with what's a good approach to address that.

So when we talk about network attacks-- not to go into detail-- there is a wide variety of network attacks out there. Every day we see news articles about new ones. The focus of this work was to look at just a couple of these. One was in this

section that we call probing and scanning, and another was in this resource usage or resource utilization area. And specifically, we looked at port scanning and distributed denial of service attacks.

As a quick example of what happens in one of these network attacks-- so this is an attack, often, within the bucket of probing and scanning. So essentially, what happens is you have an attacker that attempts to find out what ports are open on a victim or a target system. It does this by sending requests, similar to pings, to a victim or a target system. If the target's system acknowledges one of these things, you can say, oh, this particular port is open. And then, one may go and look at what software typically runs on those ports and attempt to use some of the known vulnerabilities of that software.

So a lot of software packages have specific ports that they tend to use. So you can say, oh, you know, port-- I'm making up a number, here-- port 10,000 is open, and I know that's used by Microsoft SQL server or any other piece of software. And then you can say, well, here is a known vulnerability that I could use, and then I'll try attacking with that. So this is just a simple technique that a lot of attackers use, very low bar, very easy to do. You could write one by mistake. But just to find out what ports are open, and then try to use known vulnerabilities on different ports.

Now, the least common denominator, the easiest piece of data to collect, is what's called the network packet. I'm sure many of you are familiar with the concept of a network packet. For those that are not, think of if you're sending a letter, it's all the material on the envelope. So it's the address, where it's coming from, who it's going to, plus a little bit of information of sort of what's in the package as well.

The reason that we often use network packets-- and I'll kind of open this up to the class. A lot of the cyber research actually focuses on network packets, but those often form a very small percentage of the actual data, the payloads, where actually a lot of the data actually is. Can anyone guess why we tend to use a network packets rather than payloads? Not Albert?

AUDIENCE: Header.

VIJAY GADEPALLY: Sorry, the network header. Yeah, sorry the network header, not the packet, yeah. So, Yes. Anyone guess why we tend to use the header rather than the payload of a

packet?

AUDIENCE: Encrypted.

VIJAY GADEPALLY: Yeah, encryption, exactly. So very often, especially these days, the payload of the packet is typically encrypted, so there's not too much that you can do with it. So using the header information-- the header is not, right? It's the outside of the envelope. The inside of the envelope is something that you cannot see by putting it up to the light. So that's why we tend to use the header of the packet.

So the headers-- so this is what a packet looks like, on the left. On the left side, that's what a packet looks like. And if you kind of convert that into a human readable form, just the header information, this is the type of data that you get out of it. So it gives you things that you would expect. What was the IP address that I started from? What was a port that I started from? What was the destination IP address? What was the destination port? Plus a bunch of other information about certain flags that were set, whether it's what direction to flow-- what direction the traffic is moving in and stuff like that, and then any flags associated with it, what type of packet it is, so what type of protocol is the packet using, et cetera?

All right, so just as a reminder, the data we're using is from the MAWI working group, but there is a lot of other data out there that you might be able to get your hands on. So one of the challenges, how many people here have worked in cybersecurity or done some research in cybersecurity? So a handful of people.

So one of the huge challenges-- this is something that you may run into in your data sets-- is ground truth, is understanding when-- So if you're trying to find out when there was an attack in your data, someone needs to have-- you know, there needs to be some ground truth associated with that. As much as we'd all love to sit and write DDoS attacks and port scan attacks on ourselves, sometimes, you know, getting that at scale can be very difficult.

So for the purpose of this work, we used a synthetic data generator, in which we took real internet traffic data but then injected synthetic attacks into it. And that was sort of our way of looking for-- of establishing ground truth.

So the MAWI working group actually does published ground truth based on a

number of detectors that they have. A lot of these are heuristics-based detectors. In our use and in our looking at that ground truth, it was really difficult to actually understand when an attack was occurring.

It was sort of like, somewhere in this hour an attack occurred. Which, if you're trying to train a machine learning classifier on that or an anomaly detector, that's kind of vague. Because there's a lot happening in an hour of internet traffic, and so if it's somewhere in that hour an attack occurred, that can be very difficult to find. So that's sort of a reason that we focus on using a synthetic attack generator. Yeah?

AUDIENCE: Can you say a little bit more about what kind of data gets contracted? You know, is it just additive, where you're showing some new compromised post that is injecting data to some destination? Is there also, like, return flows that are in there?

VIJAY GADEPALLY: Yep.

AUDIENCE: OK.

VIJAY GADEPALLY: Yeah, so the question that was asked is, you know, I guess, how sophisticated is this tool, and how detailed can you do it? So the exact details of that, I would probably forward you to Emily, but from looking at what was being done, these are reasonably sophisticated. So it's not as simple as, you know, you have to give it a text file, all the IP addresses, source and-- But you do have to set some parameters.

So what you can do, for example, if it's a DDoS attack-- right, that's maybe the easiest one to describe-- is you would give maybe a range of IP addresses and ports that you'd like to see injected, you'd give it a flow rate, and then you could tell it what type of packets you'd like introduced. And it sort of allows you to do a number of these different-- this particular tool at least-- allows you to pick a number of these different features that it then injects into the original pcap file.

If you're looking for some of the other attacks-- so this is just a list of the attacks that they currently support, as of a few months ago, and they might be adding to this. You know, we focus on a few of these that were also easy for us to reason and, kind of, go back into the actual data that we collected to see if the actual injection made sense to us or not. But we do have a number of different parameters that you

could pick when you're doing that.

So that's just a little bit-- You know, as we talked about, in the data conditioning phase of the labeling data, is often a pretty difficult process. In the world of cyber anomaly, in the world of network packets and network security, going back and trying to label is an extremely time consuming task and very difficult and up to a lot of interpretation as to what one calls an attack. So while we did manually inspect a few of these, and that's certainly not scalable. So we decided to go with a synthetic attack generator. And I'm sure for some of your projects, you may also kind of think about is that a route that you need to take, at least to start off with.

OK, so this is the overall pipeline of the system. So as I mentioned the first time we chatted, we spent a majority of our time in the first step here, which is data conditioning. So within data conditioning, we did, you know, taking the raw data, which is downloading this from the internet. This comes to you in a binary format, which is a pcap. For those who are familiar with packet capture outputs, that's that binary format that comes from, like, tcpdump, for example. We then convert that pcap file into flows, parse that into human readable form, feature engineer, training machine learning classifier, do the same thing with the synthetic data, and then get good results. Or we hope that you get good results.

So step one of data conditioning was to download the actual raw data. The data was downloaded in a compressed binary format, which is probably what a lot of you will get your hands on. A typical size of a single zip file or typical compressed file is about two and a half gig. When you decompress that, it goes to about 10 gigs, so about 4 x. And a single pcap file corresponds to-- so it's about 10 gigs for about 15 minutes worth of traffic.

So if you're trying to do multiple days, you can imagine how this starts to explode in terms of volume. And about 15 minutes, on average, is about 150,000 pack-- it corresponds to about 150,000 packets per second. So this a reasonable amount of network traffic that we're getting our hands on, but not even close to the types of volume that large scale providers have to deal with on a regular basis.

Once we have the actual-- once we have the pcap file downloaded, just the network packets don't really tell us much. It's stuff to do things with. So what we want to do is

convert this into a representation that's useful to do analysis. And so a lot of work has been done using network flows. And a network flow is, essentially, a sequence of packets from a single source to destination.

So as you can imagine, when you're streaming a YouTube video, for example, it's not one big packet that comes to you, but it's a series of packets. And so flow essentially tries to detect that and say, OK, all of these things was a person watching this video. Right? So from this source to this destination within some time window is defined as a network flow.

And so we convert each of these 15 minute pcap files into network flow representation using a tool called yaf, which stands for yet another flow meter. And so that helps us do some of this flow-- establishing these network flows. And so the size of about 15 minutes worth of flows goes to about two gigabytes, and we set a time out of flow to be about five minutes.

OK, once we have the flows, they're still in this binary format. We need to now convert them into a representation that we can look at. So flow comes with an ASCII converter called the yafscii-- Sorry, yaf comes with the ASCII converter called yafscii, and that essentially converts the data from this floor representation into a tabular form.

And if there's one lesson that I think Jeremy will talk to you a lot about next, it is tables, tables, tables. People love looking at tabular data, it's very easy to understand. I wouldn't recommend it, but you could open this up in Excel. It would be huge, but it would open up. But it's very easy to, kind of, look at, especially when we're trying to establish truth in the data and go back and take a look at it.

So for our pipeline, we take the yaf files, convert them into text files. Each of these ASCII tables, so again, about 15 minutes worth of data, is about eight gigabytes in size. And we record a number of different data pieces per flow. I won't go into the details, but if you're interested, you can kind of look at what are the various entities or features that come up in a network flow?

So now that we have network flows, that's sort of like part one of data conditioning, right? We've converted it into some representation that makes sense. We have it in human readable form. So, step one data conditioning done.

Now, the next thing is to convert it into something that actually makes sense for our machine learning algorithm. And so this is sort of bucketing into the area of feature engineering, when we kind of think back at it. And so feature engineering is really, really important. You'll spend a lot of time doing this on new data sets.

And so, each of these flows contains about 21 different features. And one of the-- we need to look for essentially which of these features makes sense for us to pass into a machine learning model. Others we're just going to end up training on a [? noisy. ?]

So we have to use some domain knowledge, right, where we talk to experts in cybersecurity and say, well, you know, IP address is kind of an important thing, but maybe this other flag is not as important. And then, we use a lot of trial and error and luck, as well, to help pick some of these features.

Once we have these-- Once we kind of pick a set of features that we're interested in, when we look back at the anomaly detection literature, a lot of work has been done with this concept of, you know, you have these flows, but you need to convert them into some other format that makes sense to look for anomalies. So it's just converting into a feature space in which anomalies make sense.

And so we looked at using entropy. And the basic intuition behind this is that if we're looking at various flows between different fields and that the entropy of these flows should be roughly equal-- should be roughly similar, should be about the same, not equal-- but should be unchanging when there is no big mechanism, defined by an outlier detection mechanism, that's involved with changing the entropy.

So for example, just to make this more clear, if you're having a DDoS attack, this is typically a number of different source IP addresses attempting to talk to a single destination IP address. So you would expect to see an increase in the entropy of the source IP field in that particular example. And in port scan attacks, you would probably see something on the destination port entropy would go up. Right, that's a little bit of intuition, it's a little bit more complicated than that. But that's a very high level view of what's going on.

And for entropy, we just use the standard-- I'm sure many of you who are familiar

with information theory are very familiar with Shannon entropy. And we compute the flow associated with each of the features that we pinged from the network flow before. So from the feature engineering, we picked a subset of features associated with that.

We all remember when neural networks are. This is just there for completeness. So we take these now, network flow sees these entropies associated with the network flow, we use those as input features for a neural network model. Now, you'll recall from the neural network talk that we had last week, we have inputs and outputs, we have weights associated with them, and then we have this nonlinear equation that sort of represents inputs and outputs at each of the layers. That was the equation. I could see Barry giving me the cue that I'm walking towards the slides.

AUDIENCE: Oh no.

VIJAY GADEPALLY: So the features that we have over here correspond to the various entropies that we've computed in the previous step. And the outputs of this network correspond to a class of attack. So in particular, we focused on, obviously, no attack, DDoS attacks, port scans, network scans, and P2P network scans.

The model itself had three hidden layers with 130 and 100 nodes, respectively, and an output layer. And we used ReLU activation. I'm happy to talk in more depth about why we selected these, but I'll save that for another time.

And going through this process, we actually had very good results. So that's sort of the short form of the research work. But what I wanted to really emphasize is the amount of effort that went into the data conditioning, the importance of collecting data, anonymizing data, making it human readable, converting it into a format that people understand, and then kind of walking through the sort of-- it was certainly an iterative process. As much as this is a very short form of what we did, this took a long time to get to these results.

And what we're presenting over here is the attack type and the intensity. So this is the ground truth from the synthetic data generator, and then this is sort of the prediction that our model has across the site. And this is a confusion matrix. So a higher number or a darker blue indicates that we did a really good job. And we've also used varying intensities of these in attacks. So certainly, if we have strong,

which means we've injected about 20,000 packets in the 15 minutes, we can detect those really, really well. As you have weaker and weaker attacks, our system doesn't do as well. it's quite reasonable, but certainly an area of ongoing research.

So with that, I just wanted to hand it over to Jeremy. But these are some initial work and results of detecting and classifying network attacks. The keynotes was-- data conditioning was where we ended up spending the majority of the time. Cleaning up the collected data, coming up with-- We spent a lot of time trying to figure out how to label this data because we were just unable to really do it. And finally, we ended up going the synthetic generator path. But we actually did spend a significant amount of time to try to figure that out. And then, we spent a lot of time and feature in engineering, which did consist of a lot of trial and error and a lot of that stuff like that. OK, thank you.

JEREMY KEPNER: OK, thank you very much. I'm Jeremy Kepner from the Lincoln Laboratory Supercomputing Center. I think most of you know me already. I'm just really introducing myself for video.

And this has been his topic has been teed up really well by Sid and Vijay. I think they illustrated, sort of, going from-- if we think about our pipeline going from left to right. So they've talked about, kind of, the end stage of what you're trying to do and building these models. Vijay talked about the data that feeds into that. I'm going to talk about the data architecture that you go through to kind of build up to that.

And the reason we've done it in this order, which is the reverse order that you do it in real life, is because it's difficult to appreciate the data architecture piece without an understanding of where you're going. And so if you see from Sid's talk, I think it became very clear that having lots of data is really important for creating these models. Without the data, you can't do anything. And likewise, you didn't see it, but all that data was created in a very precise form that made it easy to do that kind of work.

And then, Vijay talked a little about some of things, given a data set what did he have to do to get in and form such that it would work for his AI training application. And what Vijay is doing and did in that application, is very much a sort of microcosm of which many of you will be doing in your own projects. You will try to

identify data sets, you will try and do featuring engineering, and then go through this sort of trial and error process of coming up with AI models that suit some purpose.

And a key part of that is what we call AI data architecture. So I'm going to talk about some of the things that we've learned over the last 10 years in data architecture. As a head of the supercomputing center, I have the privilege of working with, really, everyone at MIT on all their data analysis problems, and over the years, we've learned some techniques, some really simple techniques, that can dramatically make data architecture for your applications a lot simpler.

There's a lot of people out there selling very fancy software to solve data architecture problems. We have found that we can use very simple techniques to solve most of those problems using software that's built into every single computer that is made today. That is, you don't need to go and download some new tool to do any of the things I'm going to talk about right now.

So with that introduction, I'm going to just go over this the outline of my talk. I'm going to add some introduction and motivation, which I'll go through quickly because I think Sid and Vijay did a fantastic job of motivating what we're doing. And then, I'm going to talk about our three big ideas and how to make AI data architecture simple and effective across a wide range of applications.

One is, and as Vijay talked about, we're going to really hit this concept of tabular data. There's a reason why tabular data is so good and why it is such a preferred format. And so, you want to try and get your data into these tabular formats. And the nice thing is that, if that sounds really simple, it's because it is really simple. Lots of people, though, want you to buy tools that make this process complicated because that's how they inject themselves in the process. And we've discovered, actually, you can do world class MIT quality AI work with really, really simple tools. Getting things in a tabular form is a great starting point.

And then, how to organize the data, how to organize the data in the pipeline that Vijay showed, which is pretty standard across a lot of applications. Again, there's lots of applications and tools that we'll will to do that for you. We've just found that simple folder naming and file naming schemes gets you a long way. They require no

tools, and they make it really easy for other people to join in your project because they can just look at the data, and it's pretty clear what's going on. And it also makes the data easy to share. So that's another simple thing.

And then finally, I want to talk about every single application we've demoed here involve data steps that were designed for sharing. Designing your data such that you can give it to others actually addresses a lot of the data architecture problems that you will encounter yourself. By thinking about, OK, how do I make this data such that I can give it to someone else, you're really making a much better product for yourself. You're making it so that other people can join your team much more quickly and be much more effective. And so, thinking about sharing of data is really an important thing.

And what you'll discover is that generalize sharing of data is very hard. Just pointing to a data set and saying I want to share it is very challenging. There's a lot of barriers to do that. But if you have some idea what the end purpose is, what the particular application that you want to share this data for, then it becomes a lot easier, both technically and administratively. And we will talk a little bit about some of the administrative burdens that you need to go through to share data. And this is where knowing the purpose, in this case, we want to do particular AI application, dramatically simplifies the process of doing data sharing.

So just some motivation, this is a slide that Vijay presented earlier. And the big thing to emphasize here is we have a bunch of breakthroughs, and the key is the bottom row, which says that the time from algorithm to break through is, like, two decades, but the time from getting data set that is a shareable, well architected data set to break through was much, much shorter, was only three years.

And so that's, kind of, motivation for the fact that why data is so important to making progress in AI. The data is actually writing the program, as Vijay mentioned in the first lecture. And so having data is really important. And it has been often observed that 80% of the effort associated with building an AI system is data wrangling, or data architecture, as we call it. And one of the benefits of these challenge problems, or these open and shareable data sets, is that they have done a lot of the data architecture and data wrangling for the community, which is why they're able to have a lot of people work on it and make progress on it.

If you're doing a new application, you have to do this data architecture and data wrangling yourself. And we're going to share with you the techniques for doing that effectively. And I would say, in most situations, this 80% of the effort actually is going to discovering the methods that we are going to cover in a few minutes. That is, it takes people a long time to figure out, oh simple tables are good. We can organize our data that way. And just having a good set of folder names and file names really solves most of the problem, right? That's the beauty of the solution. It sounds so phenomenally simple. How could it possibly work? That's actually-- you know, if there's one thing we figure out here at MIT, it's that simple can be very good and very powerful.

So as was mentioned before, sort of these neural networks, as depicted here, drive this space of applications. Most of the innovations we've seen in the past few years have been using these neural networks. This one slide sort of covers all of neural networks. In one slide, you have the picture, you have the input features, which are shown in gray dots, you have the output classifications that are shown in blue dots, and then you have their neural network and all their various weight matrices or weight tables in between.

If you had one goal, in terms of understanding AI, making a commitment to understanding this one slide, then you pretty much understand a good fraction of the entire field. And while you are here at MIT, please feel free to talk to any of us about this slide or these concepts and we will work with you to help you understand it.

Because anyone who's ever coming to you and offering an AI solution that uses neural networks, which is a significant fraction of them, they're just doing this. This is all they're doing. It's all on this one slide. Many of them, by the way, do not know the math that one equation here that's in this this. They have software, they play with knobs, but they don't even understand the underlying mathematics. And just understanding this math, taking the time to do it, you will know more than anyone who is actually trying to sell you AI solutions.

Now, as Sid said, he talked about batch size-- I just added batches to this figure to emphasize the fact that all of these letters are mostly uppercase. And if you want to

know one thing about matrix math, it is the easiest way to go from a vector equation to a matrix equation is to take all the lowercase letters and make them uppercase. And most of the time, that actually just works. In fact, typically what we do if you're a linear algebraic person, is we do that, and see if it breaks anything. And if it doesn't break anything, well, we assume it's right.

And all of these matrices, right, are tabular. A matrix is just a big table. It's a great big spreadsheet. So basically, all of neural networks in AI is just about taking spreadsheets and transforming them. And so, this is just motivation for why these tables are so important.

Here is the standard pipeline. Vijay talked about it briefly. And again, what we discovered is that, in most data analysis, which includes AI, this pipeline represents, pretty much, what most applications are doing. It's not that any application does all of that, but most of the steps in an application fall into this pipeline.

Basically, you have raw data, you parse it, usually into some tabular format, you may ingest it into a database. You then query that database for a subset to do analysis, or you might scan the whole database if you're doing all the files, if you want to do some application that involves all the data. And then, you analyze it, and then you visualize it. And so, this is sort of the big steps in a pipeline that we see.

And again, most applications that we'll will use three of these steps. They're not using all of them, but it's a great framework for looking at any data analysis pipeline. It's easy to use, it's easy to understand, and it's easy to maintain. And you know, we like all of those features. And I'll get into those in more detail. And then finally, I've already talked with significantly about data sharing. There's just some examples of data we've put up for sharing, the moments in time challenge, the graph challenge.

And again, creating, sharing quality data, really, it's not just a service to the community. It's principally a service to yourself. And in fact, we've worked with a number of applications and sponsors where we help them do this, and they find the data that they made shareable is instantly the most valuable data in their organization because it's shareable within their organization too. The same things that make it shareable to the broader community make it shareable within your

organization to other partners. So that's really important.

And again, as I said before, co-designing the sharing and the purpose of the data, is quite you need to do that. In order to do effective sharing, you can't just share data arbitrarily. That can be very difficult to do, but if you have a particular application in mind, sharing the data is quite simple. And I'll get into that later.

So I'm just going to, then, sort of hammer home some of these points, here. So nothing really new, here, just sort of repeating what I've already said about these things but with adding a little bit more texture, to them.

So why do we like tables? Well, we've been using tables to analyze data for thousands of years. So on the right is a 12th century manuscript. It's actually a table of pies. And if you can read Latin and Roman numerals, you already can tell it is probably, oh, there's some columns, and there's some rows. And those Roman numerals are numbers. Even 800 years later, we can still see that this is a data table. And it goes back further. You can find cuneiform tablets that are thousands of years old that are clearly written in a tabular form. So humans have been looking at data in a tabular form for thousands of years. Part of that is how our brain is actually wired.

One, the 2D projection, since we live in 3D space, is the only projection that will show you all the data without occlusion. So if you want to look at all the data, it kind of has to be presented in 2D form. We have a 2D optical system. In addition, we have special hard wired parts of our eyes that detect vertical and horizontal lines for, presumably, detecting the horizon or trees or who knows what. And that makes it very easy for our eyes to look at tabular data.

And because of this, tabular data is compatible with almost every data analysis software on earth. If I make data analysis software and it can't read tables or write out tables, it's going to have a very small market share. And this is just an example of some of the software out there that is designed to use tables. So whether it be spreadsheets or databases or the neural network packages that we've already looked at, various languages and libraries, and even things that you wouldn't think are tabular like hierarchical file formats, like JSON and XML, can actually be made compatible with a tablet format.

So I'm just going to walk through these in a little more detail. So of course, our favorite is spreadsheets. Here's an example of a spreadsheet, on the left. And what's great about it is that I have four different types of data that are very, very different, and you can all look at them and sort of get a sense of what they are. And that just shows how flexible this tabular spreadsheet form is, is that we can handle really, really diverse types of data in a way that's intuitive to us.

Some of the biggest pieces of software that are used that implement this are Microsoft Excel, Google Sheets, Apple Numbers. It's used by about 100 million people every day. And so, this is the most popular tool for using data analysis, and there's a good reason for it. And thinking of things in a tabular format so they can be pulled into this really makes communication a lot easier. And so again, just sort of hammering home this element of tables.

Within that, there's specific formats that we have found to be very popular. So CSV and TSV are just non-proprietary, plain text ways of organizing data. We highly encourage people to use them because then it can be used by any tool. CSV just stands for comma separated values. It's usually filename.csv in lower case or filename.CSV in upper case. And all it means is that the columns-- So each row is terminated by a new line, and within each row the columns are separated by a comma. That's all that means.

We tend to encourage people to use TSV, which stands for tab separated values, because we can replace the comma with a tab. And that allows us to write higher performance readers and writers. Because it is not uncommon to have a comma within a value, and so it makes it very difficult to write parsers that can do that fast. But it's OK to say to a person generating data, if you put a tab inside a value that's a foul on you. You shouldn't do that. And so that means I can write a program that can read this data really fast, and it can write this data really fast. And that's why, in the big data or AI space, people often tend to use TSV. And again, it's easily readable and writable by all those spreadsheet programs that I have said before.

Databases play a very important role as well. There's several different types of databases that are called SQL, NoSQL, and NewSQL . They're good for different things. I won't dwell on that here, but there are different types of databases. All of them operate on tables. In fact, the central object in a database is literally called a

database table. And so, databases play an important role. I won't go into detail here, but I just want to mention this.

Likewise, you've already had some demonstrations of different languages and libraries. I think everything you saw was in Python. There's other languages-- Julia, MATLAB, Octave, R-- many other languages that play important roles in data analysis. We've talked about Jupyter Notebooks, which is a universal interface to these different languages and libraries to allow you to do data analysis in addition to the various machine learning tools. We've talked a lot about TensorFlow, but there's others-- Theano, Caffe, Nvidia Digits-- all are examples of different tools for doing AI. And again, they love to ingest tabular data, and they're very good at writing out tabular data completely consistent with that here.

And then finally, I want to mention XML and JSON, which are very popular in object-oriented languages, like C++, Java, and Python. Those languages rely on data structures, which are basically just a set of fields, and then those fields can also be data structures, and this can continue on. These are called hierarchical data structures, the way these languages are organized.

And so it's very natural for them to have formats that they can just say, write out this whole blob of data, and it will write it out into a hierarchical form called XML or JSON. However, it's not very human readable. And I just told you that all the data analysis tools want tables, which means that these formats aren't necessarily very good for doing data analysis on.

Fortunately, we've discovered some ways that we can basically convert these hierarchical formats into a giant, what we call, sparse table. And sparse just means a table where lots of the values are empty. The entries are empty. And so we can just use that space to represent this hierarchy, and all the tools we've just talked about can ingest this data pretty easily.

Because tables are so popular, every single different approach or software package-- and we have a bunch of them listed in the first column, here-- assign different names to the standard ways we refer to the elements of a table. And there aren't that many. So when you think about a table, there's certain standard terms for dealing with what do we call the table? In Excel it's called a sheet. How do we

refer to a particular row? What do we call a whole row? How do we refer to a particular column? What do we call a whole column? What do we call an entry? And what kind of math can we do on it?

And so, whenever you're encountering a new piece of software-- And we deal with it all the time, but a lot of times you'll get software, and maybe it's related to data analysis seems really, really confusing. It's probably working on tables, and so all you have to need to learn is what are they calling these things. And different software gives it different names, just for arbitrary historical reasons. And so, if you can figure out what it's calling these things, then you'll discover the software is a lot easier to understand. So we definitely encourage you to just map whatever software into these concepts, and all of sudden life becomes a lot easier.

And because tabular data is used so commonly, whether it be in spreadsheets or analyzing graphs or databases or matrices, it's a natural interchange format. So by reading in tabular data and writing out tabular data, you're preserving your ability to have that data go into any application. Again, you have to be careful about people who want to write proprietary formats because they're basically, sort of, preventing you from basically working with other tools and other communities. And this is another reason why tabular data is so great.

All right, so moving on here, I wanted to just talk about basic files and folders. So hopefully, you know, tables are good. That's sort of lesson one. And we're going to talk about how files and folder names, well chosen is also good. Right? Again, this sounds so obvious, but it's really the thing that we've discovered, in working with thousands of MIT researchers over the past decade, that these are the fundamentals that help you out.

I've already mentioned this pipeline that we talked about. It's a way we organize the data. It's easy to build. It has built in tools in every single computer you have to support this to just create folders and files. That means it has no technical debt. That is, you never have to worry about if you have folders and files that in 10 years, oh my god, folders and files are going to go away. They've been around for 50 years, in pretty much their present form. We don't anticipate them changing.

They're easier to understand. I already talked about tabular file formats and how

useful they are. And we will talk about a simple naming scheme for your data files. AI requires a lot of data, which means, often, thousands or millions of files. Having good schemes for naming them really helps you and helps share the data amongst people in your team. They can just look at the folders and files and sort of understand what's going on. And again, this folder structure is very easy to maintain, doesn't require any special technology, and anyone can look at it

Getting into some more details about this table format, some sort of real, sort of in the weeds types of things that are really important. I've always talked about you want to avoid proprietary formats if you can. Using CSV and TSV are great. They might be a little bulkier. You can compress them. No problem with doing that. Then you get pretty much all the benefits of a proprietary format but in a way that's generally usable.

Within a TSV or CSV file, it's good practice to make sure that each column label is unique. So that's actually a big help. You don't want to have, essentially, the same column label or column name occur over and over again. And likewise, it's helpful if the first column are the row labels, and those are each unique as well. And just those two practices make it a lot easier to read and organize the data. Otherwise, you end up having to do various transformations.

If a lot of the entries are empty, i.e., the table is very sparse, there's another format we use, which is sometimes called triples format. Basically, every single entry in a table can be referred to by its row label, it's column label, and it's value. That forms a triple. So you can just write another file, can be a TSV file, that has three columns. The first column is the row label, the next column is the column label, and the next column is the value label. And most software can read this in pretty nice-- it's a very convenient way of dealing with very, very sparse data without losing any benefits. So that's a very good thing.

In terms of actual file naming, as I said before, you want to avoid having lots of tiny files. Most file systems, most data systems, do not work with small files really well. You do want to compress. Decompression doesn't take much time. It's usually quite beneficial. And so, it's better to have fewer larger files. A sweet spot, for a long time, has been file serving the one megabyte to 100 megabyte range. This one megabyte is big enough so that you can read it in and get high bandwidth, get the data into

your program quickly. And up to 100 megabyte, once you start going beyond that, you maybe can start exhausting the memory of your computer or your processor. So this is one megabyte to 100 megabyte range has been a sweet spot for quite a while.

You want to keep directories to less than 1,000 files. That's pretty common nowadays. Creating directories with a really large number of files, most systems don't really like that. You'll find it's also hard to work with. And so, that's something. You want to keep your directories less than 1,000. And so, you do that by using hierarchical directories.

And so, when it comes to naming files to things that tend to be pretty universal, is source and time of data. Right? Where you collected it and when you collected it are pretty fundamental aspects that you can rely on being a part of almost all data sets. And so, this is a simple naming scheme that we use that we find is very accessible.

We just create a hierarchical directory here. It begins with source, and then we have the years, the months, the days, the hours, the minutes, however far you need to go to kind of hit your sort of 1,000 files per folder window. And then, we repeat all that name within the file name itself, so that if the file ever gets separated from its directory structure, you still have that information.

And that's really important because it's easy for files to get separated, and then, you know, it's like a kid wandering around without his parents. But if it's got a little label on it that says, this is my address, then you can get the get the kid back home. And likewise, sometimes you'll do the time first and then source. It really depends on the application. They're both fine. But again, this sounds really simple, but it solves a lot of problems.

If you give data to people in this form, they're going to know what it means. Right? It doesn't require a lot of explanations. Like, dates are pretty obvious, source names are pretty obvious, and then they can, basically, work through it pretty nicely.

Databases and files can work together. You'll have some people like, we do everything with files, or we do everything with databases. They do different things. They serve different purposes. So databases are good for quickly finding a small

amount of data in a big data set. So if I want to look up a little bit of data in a big data set, that's what databases do well. Likewise, if I want to read a ton of data, like all of the data, and I want it analyzed, that's what file systems do well. So you, a lot of times, keep them both around. You have databases for looking up little elements quickly, and you have data and file system if I want to read a lot of data. So they work together very harmoniously, they serve different purposes.

And again, different databases are good for different things. I won't belabor these, you can read them yourself. But there are a variety of database technologies out there, and they serve different purposes.

So let me talk about the folder structure in a little bit more detail. This is getting really like-- It is really just this simple. When we talk about our standard pipeline, it's just a bunch of folders that we name, like we show on the right. So we have an overall folder that we usually call pipeline, and then we basically label each step with step 0, step 1, so people know which is the beginning and which is the end and what's happening in there. So we have step 0 raw, step one parse, step two ingest, step three query, step four analysis, step five is viz, if you're doing that.

And then, within each one of those, we'll have a folder, we'll have a readme, which basically tells people something about, OK, what's going on there. Those readmes are really good. Even if you never share this data with anybody else, as you get on in life, you will often be asked to revisit things, and just having this information so that two years later, you can go back and look at it, is really, really helpful. Because otherwise, you'll discover that you don't remember what it is you were doing. So you write a little readmes that give you some information about what's going on. And then, within each step, there's usually readme that talks to you about what the instructions in here. And then, we have a folder called code and folder called data. And basically, the code describes how you put data into this folder.

So in the step 0 raw, this is often the code, maybe, for downloading the data from the internet or how you collected your data. That's the code that, basically, puts the data into that folder. And then, with the next folder here, parse, this is the code that takes this raw data and parses it and puts it into this folder. And then, for ingest, this will be the code that takes the data from the parse form, sticks it in a database, and these might be the log files of the database entry. So a lot of times, when you just

data into a database, you'll have log files that keep track of what that happened. And likewise, this continues on down. So very simple.

And I would offer to you, if you someone gave you this, and you looked at it, without any documentation, you'd probably have a pretty good idea of what's going on. Oh, this is some kind of pipeline. I wonder where it begins. Probably step 0. I wonder what that is. It's probably raw data. Right? It's that simple. And this just makes it so easy. The more you can get stuff to people and they can just understand it without any additional explanation, that's incredibly valuable. And again, allows your teams to contribute quickly to projects.

And then, finally, we'll add a little thing here, which is a data file list. This is a list of the full names of all the data files because, especially on our systems, people are often processing lots of data, which contain millions of files. If you're going to be using many processors to do that, it really punishes the file system if you ask 1,000 processors to all traverse a data directory and build up the file name so they can figure out which files they should each process. Really, that's actually a great way to get kicked off a computer is to have 1,000 processes all doing ls-l on all the folders in a new directory with millions of files. Right, it will bring the file system to its knees.

So if you just have this one file that you generate once that lists all the names of the files, then every single process can just read into that file, pick which files they want to read, and then do their work, and then even know what the names to write the files are. And that's, really, a very efficient way. It dramatically speeds up your processing and keeps the pressure on the file system at a minimum.

In addition, if you just want to look at what you have, you don't have to type ls to list the directory, which might ls while. You can just look at this file and see what do I have. Right? It's very convenient to be able to know what the data sets are. And then you can go in there, and you see, oh, I see all these, you know, time stamps and sources. I know what I have. Right? It makes it really easy for people to get through this. So finally, I want to talk about using and sharing. And so, again, co-design is using and sharing together. Most data that you will deal with starts out unusable and unshareable, that is, the data is not suited for the purpose that you wanted to use it for, and you do not have permission to share with anybody. And so, if you understand the data and its purpose, you can get through both of those problems.

So, really practical aspects, here, of data sharing. We're going to talk about a few different roles that are very common in data sharing. The first is the keeper of the data. This is the actual person who has the ability to make a copy of the data. You can talk to lots of people, but until you find the keeper of the data, you haven't really gotten anywhere. Right? Because lots-- Oh, yes, we have data, and we can share it, or we can give it to you, et cetera. And there's someone who can eventually type a copy command to the data. You want to find that person. That's the person you really need to work with. They're the one person who can really, really help you because they actually have hands on access to the data. So you want to find that.

Then, you want to get a sample or a copy of the data to your team to assess. It is always good to do good opsec, which is good operational security, which means fewer accounts, fewer people having access is better. A lot of times people say, oh, here's the data. Give me a list of names that you want to have access to it.

If you provide them one name, that's often pretty easy to get approved. If you provide 10 names, now questions will happen. Why are we giving 10 people access to this data? You don't really need 10 people to have access to the data. You can just have one person get the data on behalf of the team. That's relatively easy to approve. You start giving a list of 20 names have access data, now there's all kinds of questions, and that may just end the exercise right there.

And once the question starts happening, you'll discover, well, maybe we've lost that one, and we need to move on to someone else. And guess what, that keeper, do they like you now that they've raised all kinds of questions because of something they thought was OK? Well, now we're-- No, you may no longer have a friend. Right? So you've lost a friend, you've lost a source of data, so minimizing the access to the data, getting just the people access to it who need you to make your next step is a really good practice.

I have been that keeper. I have been the person who has to approve things, and nothing makes me shut something down quicker than when someone says, well, I want 20 people to have accesses. I'm like, well, no. Zero works for me. That's a number in 20. (LAUGHS) Right? That's one of the numbers, there. How about zero?

That works for me.

So then, once you get a copy of the data, you then convert a sample of it to a tabular form, you can identify the useful columns or features in that, and then you can get rid of the rest. That's a huge service to your community is if you have a volume of data, and you know, like, all of this stuff is not going to be useful. Now you're just giving people useful data. That's really, really valuable. That's sort of a --

And since you're talking the keeper of the data, they may have the most information about what these different columns or features mean. That's not a conversation that a lot of people are going to get to have. And so, if you get to help your whole team by saying, oh we know these columns mean this. They're not relevant, let's get rid of them. That's a huge service to your community. It's less data. It makes it easier to release.

You then, maybe, will, if there is data columns or features that are sensitive in some way, there's a variety of techniques you can use to minimize, anonymize, possibly simulate, or use surrogate information to make it so that you can share that data. And there's lots of techniques, but those are only applicable when you know your application. So if I look at this data, and I see this one column that might be sensitive but valuable for my application, I can often come up with a minimization scheme that will work for that.

Because all data analysis is based on tables, which is mathematically just matrices, one of the most fundamental properties of matrices that we really, really like is that reordering the rows or the columns doesn't change any of the math. That's kind of, like, the big deal of matrices and tables. You can move the row, reorder the rows, reorder the columns, and everything should still just work, which means if you relabel the columns or leave relabel the rows, things should still just work, and you'll be able to do the science you need.

So a lot of times, when you share data with researchers, information that normally you would delete, that you would think would be valuable for someone else-- The researcher is like, I don't care. Right? As long as it doesn't change something fundamental about the structure of the data, I can work with that. And again, that's something you would never know unless you have an end purpose in mind.

So then, you want to obtain preliminary approval to take-- I have a data set, I've worked with the keeper, is it OK for me to share this with a broader set of people to make sure that they like it too? That's usually easy to do. It's just a small sample. And then, you can test with your AI users. You repeat this process until you, basically, have data. It's like, yeah, this is what we want, this is what we want to share, and then, you can sort of go through the process of how do we really get an agreement in place to do that.

You also, then, create the folder naming structure for scaling out to more data files. And then finally, what's great is if you can then automate this process at the data owner's site. It's tremendously valuable. Then you have just gifted them a data architecture that makes their data AI ready. And then, moving forward, they can now share with you, within their own organization, or with others. Right? So this is incredibly valuable. Right, we said data architecture and wrangling is often 80% of the effort. You do this, you've now eliminated potentially 80% of the effort of anyone else who ever wants to do data analysis or AI in this data. Right? This is a phenomenally huge win for anyone that you partnered with is to give them a very simple data architecture they can maintain going forward. It's a huge potential savings to them.

All right, so another role we want to talk about is the data owner subject matter expert, or SME. So we've talked about the keeper of the data. There's often a subject matter expert associated with the data that is an advocate. The keeper is often like, hey, I maintain the data. I do with these subject matter experts tell me to do. But there's someone out there who wants to help you get the data because they see it's going to be valuable to you.

And all these AI data products are generally best made by people with AI knowledge. So we are highly motivated to engage with these subject matter experts, pull them into the AI community so they can see the best practices. That will allow them to understand which data they have, how it might be useful to AI, and how to prepare it, so that's better. So whenever we get engaged with subject matter experts at other communities, at data owners, we really want to pull them into the community, have them come to our environments so they can see what's going on and help us identify new data sets and help make those data sets

appropriate for our consumption.

So one of the big things you'll run into is that there are concerns with sharing data. That is, there's a lot of confusion about the liability of sharing data with researchers. And data owners often have to think about the lowest common denominator. If it's a company, they have to think about US and EU and other requirements, all kinds of different frameworks for thinking about that.

The good news is there are standard practices that meet these requirements. And I'm going to list what these standard practices are. We have worked with many folks who have tried to share data successfully, and these are the properties of successful data sharing activities. And the activities that don't do these things often are unsuccessful. We didn't invent any of this. This is just our observations from working with many people who've tried to share data.

One, you want data available in curated repositories. Just taking data and posting it somewhere without proper curation really does no good for anyone. You want to use standard and anonymization methods where needed. Hashing, sampling, simulation, I've already talked about those, but those do work, particularly when they are coupled with access requires registration with a repository and legitimate need. So there's a data agreement associated with getting the data in some way, shape, or form. And those two together really, really work well.

And you also want to focus on getting the data to the people who are actually going to add value to it. So one of the misnomers is, oh, we'll just take data, and we'll make it available to all 10 billion souls on the planet earth, when, really, less than 100 could ever actually add value to us. So why are we going through all the trouble and the issues associated with making data available to anyone, when really we only want to get that data to people with the skills and legitimate research needs to make use of it?

And so, by setting up curated repositories, people then apply-- I want to get access to data-- they talk about what their need is. That sets, sort of, a swim lane. You can't just do anything you want with the data. When you say, I'm going to use it for this purpose, you have to stick to those guidelines. If you want to change them, you have to go back and say, look, I want to do something different. If you're a

legitimate researcher with legitimate research needs, that's usually pretty easy to approve.

And then, recipients agree not to repost the whole data set or to deanonymize it. This is what makes anonymization work. Not that the anonymization or the encryption is so strong it's unbreakable, it's that people agree not to reverse it. And the good thing about working with legitimate researchers is they're highly motivated not to break their agreements. If you're at MIT or any of these universities and you break your data sharing agreements, that is professionally very detrimental to you because it is very detrimental to the organization. No research or organization ever wants to have a reputation that it does not respect its data usage agreements. That will put that research organization out of business. So there's a lot of incentive for legitimate researchers to honor these agreements, which may not be the same for just general people who are working in other types of environments.

So they can publish their analysis and data examples as necessary, they agree to cite the repository and provide the publications back, and the repository can curate enriched products. So if a researcher discovers some derivative product-- Look, I can apply this to all the data. I'd like to have that be posted somewhere. They can often use the original repository to do that. And again, we encourage everyone to encourage people to follow these guidelines. They're very beneficial.

So I've talked about the data keeper. I've talked about the subject matter expert. Now, I'm going to talk about a final critical role, which is the ISO, the information security officer. In almost every single organization that has data, there is an information security officer that needs to sign off on making the data available to anyone else. They're the person whose neck is on the line and is saying that it's OK to share this data, that the benefits outweigh the potential risks.

And subject matter experts need to communicate the info with the information security officer. They're often the ones advocating for this data to be released, but they don't know the language information security officers. And this causes huge problems. And, I would say, 90% of data sharing efforts die because of miscommunication between subject matter experts and information security officers. Because the information security officer fundamentally needs to believe

that the subject matter expert understands the risks, understands basic security concepts in order for them to trust them with their signature to get it released.

And so, typically, what happens is that an ISO needs a very basic information from the subject matter expert in order to approve the release of information. What's the project? What's the need? Where is the data going to be? Who's going to be using it for how long? They are expecting one sentence answers to each of these. A subject matter expert typically replies with an entire research proposal.

And so, if I'm an ISO and I'm trying to assess whether a subject matter expert knows anything about security, and I gave them really easy softball questions, and they come back with something that clearly took them way more time and effort that answers none of my questions, what's my opinion going to be of that subject matter expert and how much I should trust them? It's going to be zero. And from that moment onward, I'll be polite but, you know, how about zero data? That works for me. (LAUGHS) Right? That's a good number, right? So this is really important.

And so, if there's one thing we can do-- Because very few of you are ISOs. And something that my team gets involved on a lot is to intercept this conversation before it goes bad, is we can help you work with ISOs, work with people who have to release with data, and help answer these questions in a proper way. And so, I'm going to go through a few examples, here.

So, a really common question is, what is the data you're seeking to share? So this is a very common question to get back from an ISO. Describe the data to be shared, focusing on its risk to the organization if it were to be accidentally released to the public or otherwise misused. Very common question.

And so, here's an example answer. Very short answer. The data was collected on these dates at this location in accordance with our mission. The risk has been assessed and addressed by an appropriate combination of excision, anonymization, and/or agreements. The release to appropriate legitimate researchers will further our mission and is endorsed by leadership. Very, very, very short answer, but we've covered a lot of ground in that short answer. And I will explain.

Sentence one establishes the, identity finite scope, and proper collection of the data in one sentence. Sentence two establishes that risk was assessed and that

mitigations were taken. Sentence three establishes the finite scope of the recipients, an appropriate reason for release, and mission approval. We've covered-- we've answered nine questions in three sentences at the level of detail an ISO wants.

An ISO can always ask you more questions for more detail, and they're usually looking for another one sentence answer. And you build these up to be three, four, five sentence things that cover broadly what you're wanting to do. And you see here, you don't want to do anything that requires the ISO to have detailed technical knowledge of what you're doing. They're looking for overall scope, overall limiters on what you're doing so that is somehow contained to a reasonable risk.

Another question, here, that's very common is where, to whom is the data going? So please describe the intended recipients. So an example answer is the data will be shared with researchers at a specific set of institutions, that it will be processed on those institution's own systems, meaning their institutions security policies, which include password controlled access, regular application of system updates, encryption of mobile devices, such as laptops, all provided access to data will be limited to personnel working as part of this effort. Again, we've covered a lot of ground very simply. We didn't really go into too many details that are specific to this particular case.

And then, a final one is what controls are there on further release, either policy or legal? So an example is an acceptable use guidelines that prohibit attempting to deanonymize the data will be provided to all personnel working on the data, publication guidelines have been agreed to that allow for high-level statistical findings to be published, but prohibit including any individual data records. A set of notional records has been provided that can be published an example of the data format but is not part of the actual data set. The research agreement requires that all data be deleted at the end of the engagement except those items retained for publication.

I'm not saying you have to use this. But again, this is another example of what ISOs are expecting. You give them language like this, their confidence swells. They're like, ah, I'm dealing with someone that understands the risks I'm taking on their behalf. And we can move forward in a very productive way. And if you ever have

questions about how to do this, feel free to talk to my team or other people or other ISOs who've had to deal with this. We actually are working with a lot of people on this language, a lot of people the government, to sort of get these best practices out there. We have socialized this with a fair number of experts in this field, and there's general consensus this is a very good method for helping you get data out.

So that brings me to the end of the talk. So as I said before, data wrangling is very important, significant part of the effort. Analysis really relies on tabular data, so that's a really good format. You can do it in very simple, non-proprietary formats. You're going to have to name your folders and file something, you might as well name something that's really generally usable. And then, finally, co-designing, the sharing, and what the intent is at the same time, that's doable. Generalized release of data is very, very hard. Tailored release for specific applications is very, very doable if you have good communication from all the stakeholders involved.

So these are just really practical things. This isn't, sort of, fancy AI. This is, sort of, bread and butter, how do you actually get data analysis done. And with that, I'm happy to take any questions.