Module 8
Video Transcripts

Video 1: Probability vs. Statistics: Part 1 (4:13)

Welcome. Today we'll be talking about probability versus statistics. We'll start off with the definition of probability versus statistics, and then we'll look at an example of statistics, an example of probability, and then we'll wrap up. So, what is the difference between probability and statistics? From a definitional perspective, we can think of probability as predicting the likelihood of future events and statistics as analyzed in the frequency of past events. So, statistical theory really does measure the extent which the world is ideal that is, what has happened. And probability theory enables the discovery of the consequences of the ideal world or that is, what may happen. And they're both very interesting for us to be able to study.

So, within Python, let's take a look at how we would create or measure statistics within Python. Let's take the coin toss example as our example because it's an easy one for people to understand. So, with probability, we seek to answer the question: what is the chance of an event happening? An event is the outcome of interest. And in this particular case, the event of interest is whether or not we have a heads or a tail. So, these two events of the set of possible events are referred to as the sample space. So, to calculate the probability of an event occurring, we sum how many times an event of interest can occur and divide it by the sample space value. So, in essence, what we're going to end up here is that we'll come to the point where we see that a fair coin, or a coin that is unweighted, will have a 1-in-2 chance of being a heads or a tails, or basically a 50-50 outcome, which we would expect.

So, let's get to some Python code and take a look at this. So, the first thing I want to do in order to be able to simulate this statistically, because we know from probability that we are expecting there to be a 50-50 chance, but in reality is that the case? So, in Python, we're going to create two functions. The first function's going to be called a coin trial, and the second one is going to be called simulate. So, coin trial allows us to be able to effectively determine the number of heads or tails that are being flipped a certain number of times. In this particular case, we're going to use 100 attempts. And then we're going to measure whether or not the random function generates a value that is less than .5 or greater. And then if it's less than .5 or equal, we're going to increase the number of heads we'll assume that that's a heads and if not then by default it's a tails. And then we're going to create a simulate function which allows us to be able to invoke the number of trials x amount of times, and then be able to sum that, and divide it by the frequency with which we were able to run the test. So, in essence, what we would expect is that based on the number of times we run the situation, we would be able to determine what is the percent of heads that are being produced? And we would expect that value to be somewhere near 50%. So, let's run this, and here I've just created the invocations of the functions with multiple values. So, here we have 10, 100, 1,000, 10,000, and 100,000, and we would expect them to be hovering around 50%. So, we can run this simulation. And notice here in the first case we have 52 because there aren't as many trials that we're actually running. But then we'll notice as we get to higher numbers, we'll tend to gravitate more consistently toward 50%. And even 10,000 is pretty close because it's hovering right around 49.95, which is very close to 50%. And so here we can see we're a little bit on the light side and everything's

49%. But notice there's a little deviation here at 100. And then just for the sake of completeness, we'll run it one more time. And then we can see again, every time we ran through this, hovering around the 50% mark.

****

Video 2: Probability vs. Statistics: Part 2 (9:27)

Next, we'll talk about probability in Python using our coin toss example. So, the binomial distribution model deals with finding probabilities of success of an event which has only two possible outcomes. So, in this particular case, this is a perfect fit for us because we want to determine whether or not we have a head or a tail which are our two outcomes. So, we can use a binomial distribution to determine that probability of finding exactly n number of heads by repeating the coin toss count number of times.

Let's take a look at what this would look like. So, first we want to actually go through the simulated or called the statistical method to be able to determine a simulated coin toss. So, this is very much similar to the example we just went through. So, I'm going to go through this a little bit quickly. So, here I want to define a function called run simulated coin toss which allows me to be able to run coin tosses or x number of coin tosses for x number of trials or n number of trials. And here, basically all this does is similar to the function we just discussed earlier, is use the random function to be able to determine a range of values and then measure that to determine whether it's a head or a tail, and it returns a number of heads. Let me just go ahead and run that to create our function. We're going to ignore any warnings that may occur. And then I want to actually be able to print this because we didn't see this graphically.

So, I want to be able to show it to you graphically. So, I'm going to call the function called simulated coin toss, and then give you a feel for what it looks like when we select n number of heads for a particular run sample size. So, let's go through that process. So, first we're going to call run simulated coin toss. In this particular case, I'm going to run it with, and I'm going to start with a much smaller number just to give you a feel for what this would look like and this is, again, the statistical instance of this. So, I'm going to run simulated coin toss for 100 trials and remember I want to allocate 10 different tosses per. And so, what I'm looking for in this particular case is it's going to give me, in essence, the histogram of what it would look like for how many different heads or tails came up during that 100-trial coin toss example experiment. So, then I go ahead, and I can plot this. And let's just go through the process and show you. So, in this particular case, if we look at the example, you can see here that during those series of trials, what I was able to come up with is the number of times out of 10 out of the 100 trials that I ran, the number of times that I got a 3, for example. It looks like approximately nine different times. And then something like a 6 for example is a little bit over 25, probably around 27 or 28 but notice that there are some gaps here. So, I've got 2, 3, 4, 6, 7, 8 and 9 and so I'm missing 10. So, there is a little bit of disparity here in how this is being run. So, this would lead me to believe that I probably need to increase my sample size or the number of trials to be able to get a much smoother looking histogram. So, let's go ahead and do that. So, let's increase it to about 10,000. Let's rerun it. And here now we get a much smoother distribution.

So now I can see what I would expect which would be a little bit more of a normal binomial distribution, would be something that would tell me that so for example, a 1 or a 2 or even a 6, I would actually have values appearing for each one of those. So this looks like the kind

of distribution that I would expect. So when you're running a sample statistically, you need to make sure that you increase the sample size or the number of trials that you run enough so that you get a real a truly good representation of the data set.

So, let's move on then to the actual binomial function. So, here I want to plot the binomial distribution just to allow us to be able to compare it to what we were just able to see using statistical analysis. So, the first thing I want to do is from the scipy.stats module is to import binom which is our binom function. So, the first thing I want to do is, again, set my range. So, this is the range which allows me to be able to set the zero through 10 different coin tosses that I'll be using. I want to set the number of trials. In this particular case, I'm just setting it to 10 trials. And then I want to set the probability that I would receive a heads. And we know from doing our analysis that it's about 50%. Well, it's actually 50%. So, then I can go ahead and start my plotting. So, I create my figures and my axes. I create my labels. And then the last part of this is that I want to actually call. And if you remember or maybe you don't remember but the binomial function has what's called a probability mass function. So, the probability mass function is going to define a discrete set of outcomes that are going to be returned in terms of the probability that a particular likelihood of an event would occur. So, the probability mass function is something that is really important from the binomial which allows us to be able to then plot the likelihood that I would receive a specific number of heads on a particular set of trials.

So, I'm going to use the PMF function here. And I'm going to use that to be able to plot my binomial function or binomial distribution according to the parameters that I've set. So, let's go ahead and run this. And I don't really do this but let's just go ahead and show what this would look like. And then here you can see I've got a similar distribution to what we were able to see statistically, but this is the probability mass function allowing us to be able to look at the result. So, let's just compare the two a little bit. So, if I look at for example let's just say the 50% range or the number of 5s that would occur. So, if I were to say perform, in this particular case, 10 trials and I was able to receive how many 5s? I was able to say I get five 5s 25% of the time. So, let's go up and look at our statistical example. And notice here this is the frequency or the number of trials that I've run. So, out of my 10,000 trials, I would expect about 25% of the time I would have received five 5s, for example. And then here you can see that looks exactly right. So, I have 2,500 of my trials where I've received five 5s, which is what I would have expected which according to our binomial distribution and the probability is exactly the same. So, this gives me a means to be able to compare the difference between statistics and probability and allows us to be able to understand the benefits of both.

So, the last thing I want to show you is that, now that I have my probability mass function, I can actually use that to perform calculations. And so, I can actually run through the calculation by hand. So, here I can specify the number of runs that I want to be able to make, in this particular case 10,000. And let me just kind of put this on a different line so that you can see. And so here I can use my binomial function again. But in this particular case, I'm going to then validate that if the probability of, say, running or receiving six different, or sorry receiving a heads up six times, I can actually perform that calculation based off of the number of runs. And then I can look at the statistical probability of that occurring. So, let's run this. And then here you can see that the probability of getting six heads for the number of runs, which is 10,000 which is what we used earlier as well statistically, is about 20%. So, let's just take a quick look at that. So, from the number of heads on the binomial distribution, it looks exactly right, 20% which I would expect. And

then let's come up here to six on the statistical and it's about 2,000 which is 20% of the 10,000 runs so everything checks out. So, that's how you can use Python to be able to do both statistical and probability calculations that can benefit you in understanding the difference between the two. So, from a wrap up perspective, we talked about the difference between probability and statistics. We talked about solving a problem using statistics. And then we talked about solving the same problem using probability and then comparing the two. Thank you so much for your attention.

****

Video 3: Sampling (8:29)

Welcome. Today, we'll be talking about sampling using Python. From an outline perspective, we'll talk about what sampling is, we'll talk about simple random samples, and then we'll talk about stratified random samples, and then we'll wrap up. So, first of all, what is sampling? So, sampling is when you can't everything enough data from an entire population and you want to be able to find a representative sample that can be used to represent the entire population. So, even with relatively small populations, the data can be needed urgently, and you didn't have time to collect all the data that you requires, so you may use a sample.

So, probability sampling is based on the fact that every member of a population has known and equal chance of being selected. For example, if you had a population of 100 people, each person would have a one out of 100 chance of being chosen in order to be able to participate in the sample set. With non-probability sampling, these odds are not equal. For example, a person might have a better chance of being chosen if they live close to a researcher or have access to a computer. So, you want to make sure that, when you're performing a sample, you really understand the type of sample that you're going to be able to use. So, probability sampling gives you the best chance to create a sample that's truly representative of the population, which is really what we're after, especially in data science.

So, let's talk about simple random sampling. So, simple random sample, every member of a population has an equal chance of being included in the sample. For example, a teacher puts students' names in a hat and then chooses them without looking to get a sample of the students. Why is that good? Because the random samples are usually fairly represented, and they don't favor any member. So, that's usually a good thing. So, we'll talk about how we do this in Python. And so, there's a function called random sample, which allows us to be able to select the number of items that we want to choose from a particular random set. Let's take a look at this. So, the first thing we want to do is import the random library, and when we import the random library, we'll have the ability to then use the function, which allow us to be able to select a random sample.

So first, let's create a list of items. So here, you can see we created a list of five items, and then based on those five items, I want to be able to pick a sample set of three. And so, this would be a random sample of three items that are going to be selected. So next, I want to change it up a little bit and then show you what that would look like as well. You can see that it's a random sample of the number of items that I want to be able to select. These are relatively small, but you get the idea. So, let's go ahead and run that. So here, you can see that I'm choosing three random items from my list, and you can see what those items would be, and they look pretty random from what we can tell.

So, I also wanted to show you what this might look like if we're using a set. So, if we're looking at, say, weights, and weights being how much somebody weighs, in a set. So, here I've got six items in the set, and I can do the same thing. I could use random.sample function, and I just specify, now in this particular case, a set, and I can select four items from that set. And when I run that, same kind of thing. It gives me a set of four items that would be selected from that particular set, then likewise, if I have a dictionary, I can do the same thing. So, if I create a dictionary, let's say the dictionary contains, in this particular case, four items. I can pick two of those items at random from the four by using the same function. And so, the key here is that, no matter what data type I have, I can use the random sample function to be able to select the number of items that I'm interested in. So, let's run that as well.

So, the next one we'll talk about is a little bit more interesting. This is called stratified random sampling. So, stratified random sampling ensures that the subgroups, or the strata, that we have available, are represented adequately within the samples that we select. So, this is really important if we have a number of different groups and we want to be able to represent within the samples that we select. So, we don't want to not or you don't want to ignore any of the groups. And so, the stratification allows us to be able to define the strata so that we make sure that we have representation from each. An example of this would be, let's just say that there's 100 students that are in a student council, and we want to ensure that we get samples from each one of the strata, in this particular case, freshmen, sophomore, juniors, and seniors, and we want an equal representation from each. So, this is good because it guarantees that members from each group is represented in the sample, and it's especially important if we can't get the entire distribution of the population represented.

So, let's take a look at what this would look like in Python. So, in this case, there isn't a simple random function which allows us to be able to do this, or the random sample function we used before. But in this case, the stratification is really more inclined to be represented as part of model selection capability.

So, as we're training models, we want to be able to be able to stratify the data set, especially in the context of doing things like training and testing a model. So, you'll notice in this particular case, we're going to use an sklearn function within the model selection library. So, it's called StratifiedShuffleSplit.

So first, I'm going to define an x and a y set of data. So, in this particular case, I'm defining a set of list items, which you can see here, as x, and then I'm going to define a set of y values. And in this particular case, I wanted to identify really just three different stratified values, so 0, 1, 2, and 0, 1, 2. So, you really only get three unique values here. You'll see what that means in a second. So, if I define the StratifiedShuffleSplit and I pass my data in, then I can actually take a look at what it would look like. And then the last part of this is that I want to actually go through once I've actually performed those splits. I can see what values were returned with respect to my training data and my testing data. So, if I run through this, which I'm going to do right now, we'll take a look at the result set, just to see what it would look like.

So in this particular case, the thing to note that's probably most important to what we're looking at, is the fact that, as I mentioned before, I'm stratifying across these three values, the 0, 1 and 2. So, I want to make sure I have an equal representation, both in my training and my testing data set. So notice here, these are the values, the 2, 1, 0; and then the 0,

1, and 2. So, I'm getting an equal representation, even though they're in different orders, of the data that would be required in order to be able to represent the stratification across my data set. So, I'm getting an even distribution. So, that actually works pretty well across all of these. So, notice the exact same thing in my second training set. So, I'm getting an index of a 1, 0, 2.

In this case, I'm getting an index of a 1, 2, 0. Again, fair distribution across the stratifications that I have. So, that actually works really well. So, that's a good example of how you can use stratification, especially when you're training a model, which is really an important characteristic of model training, but also model testing.

So, let's wrap up. So, what did we learn? So, we talked a little bit about sampling and what it was. We looked at sample random samples, and then we looked at stratified random sampling in the context of model selection, model training, and model testing. Thanks for your attention.

****

Video 4: Random Variables (8:18)

Welcome. Today we'll be talking about random variables. From an outline perspective, we'll talk about what a random variable is. We'll talk about discrete variables, continuous random variables, and then we'll wrap up. So, what is a random variable? A random variable is a numerical description of the outcome of a statistical experiment. A random variable that can only assume a finite number or an infinite sequence of values is said to be discrete. And then one that can assume any value in some interval on a real number line is said to be continuous.

Let's look at this a little bit further. So, a discrete random variable is one that takes on only a countable number of distinct values. Thus, it can be quantified. So, in this particular example that we're going to give, if we roll a dice, we would say that X can only take on the values 1 to 6, and therefore it's a discrete random variable. A continuous random variable is slightly different. So, a continuous random variable is one that takes on an infinite number of possible values. And as an example, you can define a random variable X to be the height of students, say in a class. Since a continuous random variable is defined to be over an interval of values, it's represented by the area under a curve or an integral. What that basically means is that the heights of the students is never the same -- well, depending on the granularity at which we measure it. But you can imagine, in the particular case of a continuous along the line, that you may not get the same value twice.

So, now let's talk about discrete random variables. There's a base class called rv discrete which we're going to use to construct a specific distribution class. This will allow us to then be able to create a distribution which will allow us to be able to define the set of discrete random variables that can be used as part of a function. So, let's take a look at this in more detail.

So, the first thing we want to do is import numpy, import scipy the stats function. Then we want to define a set of values that we're going to be able to use. So, we're going to say that there are going to be seven values, and then we want to assign those values in this particular case to the function's x. So, we have distinct values which are going to be assigned to each one of the values that we're using.

So, in this particular case, you can see pk are those values. Then what we're going to do is we're going to use the class to be able to define a discrete function which we're going to call custom which will contain the values from x so our seven values and then we're going to assign them the specific values from pk. Then let's just take a look at it and then we could plot it. So, here I just want to show, show that function that we created. And let's run it quickly. And here you can see that because there's only a specific set of values for each one of these, we're only going to show one of a very particular set of values coming from the set that we've defined. So, next we want to talk about continuous random variables. So, continuous random variables are a little bit more difficult, especially from a class perspective. So, the class rv continuous is also a base class which we can use to construct a distribution class, but it can't be used directly as a distribution. So, in this particular case, we're going to use a histogram to demonstrate a continuous distribution. So, the first thing I'm going to do is I'm going to use the class function and define my own gaussian distribution just to give you a feel for what it would look like if you were to create one on your own. So, in this particular case, I'm going to create a class called gaussian gen. And then here, I'm going to make it a class type rv continuous.

And so, what I'm going to do is define a function called underscore pdf, which is my probability distribution function. And I'm going to return the actual calculation for the gaussian distribution for a particular x or for any x. So, once I define that function, then I can actually use it to be able to define what my value would be or my probability would be for a particular gaussian probability for a particular x or a loc, and a scale. And so, let me just run this here after we define the function to give you a feel for what that would look like. In this particular case, for the gaussian function that I've defined, the probability here of x being equal to 2.1 is equal to about 9%. I'll show you this in a little bit because we're going to use the histogram in order to give you a better feel for what it would look like graphically. So, let's go on to that. So, this is the graphical representation of it. So, in this particular case, as I mentioned, I can use a histogram because a histogram is also a continuous function and can represent a continuous function. So, in this case, I want to create first a data set. So, I'm going to use a normal distribution and I'm going to use 100,000 points. I'm going to use a loc of 0, scale of 1.5, and I'm going to create my data set based on this function from scipy.

Then I'm going to create my histogram by using the data that I've just created. And then I'm going to create what's called histdist, and this is going to be my histogram function which is going to contain the pdf and actually a cdf function which will allow us to be able to see the probability distribution function. And I'll actually use that later to be able to show you what the values look like. So, let's go forward a little bit. So, the first thing we want to do is we want to look at the histogram distribution for a particular x. So, let's take we need to assign an X value. Then we're going to look at a particular range from minus 5 to 5. And then I'm going to go ahead and plot this. So, let me plot dot show this just to give you a feel for what this would look like. And you can tell that this is a normal distribution. It's actually going from minus 5 to plus 5, as we defined. In the histogram, there are a lot of points in the histogram so it looks a pretty smooth line here, but you can see a little bit or rigidness.

So, now because I've created this, and now I have the probability distribution function that I can use to be able to see the continuous value change. Because now we are in a continuous random variable set, I could use any number of different inputs for X and then be able to calculate the probability that that would occur because they are not discrete. So, in this particular case, we're going to use a series of values 2, 2.1, 2.25, and 2.3 just to see the

different probabilities that we would see returned. And so, here you can see some of the distinction between the different values. And we could go finer levels of granularity would probably require higher degrees of precision to see. But you get the idea that, in this particular case, the probability distribution function is continuous, and it allows us to be able to see the probability for a wide range of values based on the granularity that we're interested in.

So, from a wrap-up perspective, what did we talk about? We talked about what random variables are. We talked about discrete random variables, and then continuous random variables, and we showed a few functions that we could use to be able to not just graph them but also to create classes where we defined a discrete, and a continuous random variable function. Thanks for your attention.

****

Video 5: Probability Distribution Functions (8:23)

Welcome. Today, we're going to be talking about probability distribution and density functions. From an outline perspective, we'll start with a definition, then we'll talk about discrete probability functions, continuous probability functions, and then we'll wrap up. So, what is a probability distribution and density functions? And we need to talk about them together, and I'll explain why in a second. So, a probability density function is a function that may be used to define the particular probability distribution.

So, what do I mean by that? So, if we look at the graph here, we can see that the probability density function allows us to plot f of x, which is effectively the it's the line that defines the distribution that we'll be using for determining the probability of a particular sample x. So, the probability distribution function allows us to be able to take a look at what is the likelihood that x falls within certain probabilities? In this particular case, between a and b. For a discrete function, this allows us to be able to define exactly what that x will be. For our probability function that is continuous, we are looking at it from the perspective of, what is a continuous value? So, it's going to be changing not changing in terms of the actual probability but changing in terms of the points along the probability density function line. So, what I want to do next is talk a little bit about discrete probability distribution in terms of, how would we look at those distributions within Python, and I want to look at them graphically. So, discrete probability distributions, there are three types that I'll be talking about.

There are many more, but three examples are binomial, the Poisson, and then Bernoulli. And so, in order to be able to define those, the first thing I wanted to do is show you some code. So, let's take a look at what this would look like in terms of being able to create a binomial distribution within Python. So, the first thing I'm going to do, is I'm going to include my Seaborn library.

I'm going to define a binomial function, so I'm going to collect a set of data, and generate a set of points, which allows me to be able to show you what the binomial distribution would look like. Then I'm going to create my matplotlib, or pyplot axes, I'm going to set some labels, and then I'm going to show a particular diagram. So here we go. This is what it would look like. This is our binomial distribution. So, this is what you can expect in Python if you're going to graph a binomial distribution.

The next one I want to walk about is called a Poisson distribution. So, the Poisson distribution is a little bit different. Poisson distribution is a little bit left leaning in terms of the way it's defined. So, let's take a look at how we would incorporate that directly within Python. So here, I'm going to use what's called the random function from NumPy, and if the random function is going to be a Poisson distribution, we'd have a series of points. And then, like I was doing before, I'm going to actually show the plot directly within matplotlib.pyplot. And looks like I forgot an "I" there, so let's go ahead and run this. And you can see the graph is exactly what we expect in terms of Poisson distribution. So, this gives us not just the density function but also gives us the actual distribution. So, we can look at any particular value, and then we could look up to see what the probability is in terms of the frequency, which is on the y-axis.

So, the next one I want to cover is what's called a Bernoulli distribution. So, the Bernoulli distribution is a special case of a binomial distribution. And effectively, what this means is it really just has two different plots, so two different areas that we want to be able to measure. So, let's take a look at this, and Bernoulli distribution is defined in SciPy.stats, so another library that we're going to use. We would define our Bernoulli data by calling the Bernoulli function, and we're passing in the amount of data that we want to be able to generate. And then notice that we'll bucket it into two particular buckets here, and then we're going to go ahead and show you what the plot would look like. So here you go. This is exactly what we would expect in terms of Bernoulli distribution. And so, this is the type of distribution that we would use to define a discrete probability where we only have two potential values, two outcomes. Next, we're going to talk about continuous probability distribution, and continuous probability distribution determines the possibilities or possible values from a continuous random variable, meaning that it's not a discrete set of variables, but basically falls along a continuous line. So, in this particular case, we can pass in just about any variable and expect our probabilities to change based on the probability density function that's associated with the particular distribution that's continuous. So, in this case, I want to describe just a couple of them. So, two of them that we're going to describe, the first one's going to be a Normal distribution. The second one's going to be Chi-squared.

So, let's take a look at a normal distribution in Python. What would that look like? So, in this case, I want to include my pyplot library. NumPy is going to be the library that I'm going to use to be able to create the data set that I want to be able to use. So, I'm going to use a normal distribution that we're going to create. So here, I want to use the standard deviation of 2, and 10,000 points. So, I can set that up, define my variables for my mu, for my sigma, and then I can call the NumPy function, which is a random normal function, pass in the parameters, and get back my set of points. So then what I can do there is I can describe this in terms of a histogram, and so I can plot a histogram, which would define each of the buckets where I want to collect that data, and then I can plot it. And if I plot it, I want to be able to define it in terms of not just a histogram points, but I also want to draw a line, which allows me to be able to see those values in more smooth way, or, in essence, what I'm drawing is my probability density function. So, let's go ahead and run this, and you can see here, it looks like a normal distribution. Not only do the buckets that I've created demonstrate a normal distribution, but also the probability density function, which is indicated here in yellow. So, the last one I want to talk about is called a chi-squared distribution.

The Chi-squared distribution is really just a statistical test method which allows me to determine categorical values that have a significant correlation between them. And so, the variables should be from the same population if we're going to use this. For example, things like yes or no, male or female, that is, in terms of the categorical nature of them. So, what does it look like in terms of being able to plot chi-square distribution? So in the case that I'm going to use, or give you, we're going to look at this in terms of four different lines, and the four different lines are going to have different degrees of freedom as we go ahead and plot this. And so we're going to use NumPy once again, and pyplot, which we are going to use for our graphing, and we have four different line styles, which I want to be able to describe for the different types of Chi-squared distributions. So, here's where I start to describe the degrees of freedom. As I get into my plotting, I can call from a stats package here the chi-squared probability density function.

Notice the probability density function, and then I continue on, and create my labels so that it looks nice as we take a look at what the chi-squared distribution would look like. So let's go ahead and run this, and you can see here, I've got my four different plots all on one diagram, which allows me to be able to see for each one of the different degrees of freedom that I've been able to pass in, I can see the frequency or the values that were expected from the probability based on the particular value or continuous values that I may use.

So, let's wrap up. So, what did we talk about today? So, we talked about the probability distribution and density functions, what they were. We went through discrete probability functions, and continuous probability functions, and, in the end, we learned about how to graph them. Later in these sessions, we'll talk a little bit more about the actual usage of the probability density functions themselves. Thanks for your time.

****

Video 6: Probability Mass Function (7:14)

Welcome. Today we'll be talking about the probability mass function. From an outline perspective, we'll start with the definition of probability mass function or PMF. Then we'll talk about discrete variables and the PMF, and then PMF using binomial, and then we'll wrap up. So, what is the definition of a probability mass function? So, it's a function that gives us a ability that a discrete random variable is exactly equal to some value. This differs from the probability density function in that the probability density function is associated with continuous variables rather than discrete random variables. So, let's take a quick look. So, in this particular diagram, it's indicating that we have values of x which are equal from 2 through 8 and then the probabilities are given here as the probability as a function of x. And so here are the potential values in this particular case so it's relatively uniformly distributed.

Well, let's take a look at discrete variables and the probability mass function itself. So, in order to describe PMF, let's take an example. So, in our particular example, we're going to use rolling of a dice or a die. So, we want to describe the random variable that corresponds to the outcome. In this particular case, the outcome can be any one of the values from one to six, which are the faces of the die. So, the aim of the probability mass function is to describe the probability that we will obtain each one of the values. In this particular case, we would say here that the probability of any one of those faces is going to be equal. The probability of x equal to 1 or 2 or 3, etcetera are all the same.

Since we have six possible outcomes and they all have the same probability, we would say the probability of all of them being equal is equal to 16. So, let's take a look at what this would look like in Python. And here we're going to use statistical measures to be able to determine what the probability mass function would look like. So, let's just set this, let's set the example up. So, in this particular case, we're going to say the number of throws is going to be equal to a relatively large number, we'll change it in a second, equal to 100,000. Then we want to measure whether or not any one of the outcomes may have occurred, and so in this particular case we're going to use a random choice between one and six. Then we want to measure the outcome. Then we want to go ahead and graph it. So, the last thing we want to do is we want to show this. That's fairly simple. So, let's just go ahead and run that. So, after we run it 100,000 times, we end up with approximately a probability of .17 that any one value may occur so that's probably what we would expect.

Actually, it's exactly what we expect. They'll start to gravitate more toward approximately .17. So, let's change this up a little bit just to see what it would look like. So, if let's just say I ran I threw the dice 10 times and I reran this function. You'll see a little bit more of a distribution here because the likelihood that I'm going to get a particular value is going to be much different. So, in this particular case, I only got a 2, a 3, a 4, and a 5, and so I didn't actually do so good in terms of determining the actual statistical variance of the values. So, let's put that back up to 100,000 and rerun it. And here you see what we would expect, a more uniform distribution.

Next, we're going to talk about the probability mass function using binomial. So, here we're going to use a Python library to help us. So, we're going to use the stats package which are alluded to here. So, we use the stats package, binomial library, and then the PMF function. So, what we need is a couple parameters in order to be able to use that probability mass function. So, here we're going to set a p-value to 0.5 which is the last parameter. We're going to set the number of trials equal to 10, and then we're going to specify the number of tries that we would take here. It's also going to be between zero and 10. And so if we run this function, you can see that the probability that any one of these values coming up is going to be equal to what you can see here in the returned array. So, it's a little bit hard to read this, not too hard, but we would like to see especially for a binomial function, what would that look like graphically? So, I created a small function which would allow us to be able to do that. So, here we're going to pass in some parameters.

We're going to pass in the number of tries that we want to take and then we're going to pass in the p-value as well. So, we'll just keep using a p-value of 0.5. And then we want to create a plot by calling the again, calling the binom function for the probability mass function that any one of the selected criteria will come up. Okay. So finally, we want to be able to show using Matplotlib at the end. So, first we create the function. So, let's just go ahead and do that. And there is nothing to see because we just created the function. So, the next part is really where we get into calling the function with a particular set of values. So remember, as before, we wanted to be able to call a binomial PMF function to be able to determine so for each particular value that we may select, we want to be able to see what is the probability of it occurring using a binomial function? So, let's run this. And it's pretty easy for us to see. So, the likelihood of zero is zero. And then the likelihood as we look little bit further of a five is equal to .25. So, that gives us a really good distribution of what it may look like for any particular value that would occur of the 10 selections that we have available. So, let's just increase this maybe to, say, 100 values.

Now if we increase it, we should see the same binomial, binomial distribution but I think our x-axis will get a little bit complex so let's just go ahead and run that. So, here we can see same binomial distribution, but we can see but notice that the x-axis is still kind of complicated at this particular point. So, the probability of achieving any of these values here is based on the binomial function is depicted by this graph. Let's just put that back to 10, make it a little bit easier to see, and then that gets us back to what we would expect to be sort of a normal probability mass function.

So, what did we talk about? So, we talked about the definition of a probability mass function. We talked about discrete variables and the probability mass function. And then we talked about the probability mass function using a binomial. Thank you for your attention. Thank you for your time.

****

Video 7: Cumulative Distribution Function (8:13)

Welcome. Today we'll be talking about the cumulative distribution function, or CDF for short. From an outline perspective, we'll define what CDF is and then we'll determine P values from the CDF, and finally we'll wrap up. So, what is a cumulative distribution function? So, it gives us the probability that a random variable X (capital X) is less than or equal to a certain value, which we'll define as x. So, the formula is the sum of all the outcomes less than or equal to x, that is defined as the cumulative distribution. So, let's take a look at this from a graphical perspective. So here, you can see on the left we have the probability distribution function. And so, if we were to define an X by the dashed line, we would want the value or the area under the entire curve, which would give us the cumulative distribution.

The cumulative distribution function itself is defined as a value between 0 and 1. So here, the threshold value is also defined by the dotted line that then tells us what the distribution value would actually be. So, let's take a look at what the function would really look like. So, the function itself is for a continuous random variable X. We express the probability that X doesn't exceed (capital X, that is) a value of lowercase x. And so, this is, in effect, we're determining the probabilities in a cumulative or sum fashion, which is in essence, in the probability distribution curve, it's the area under the curve itself. So, let's take a look at how we would actually calculate that in Python.

Next, we're going to talk, about how to derive p-values from a cumulative distribution function. So, first we're going to start with a t-distribution or a t-statistic with 25 degrees of freedom. Then we're going to use that to be able to derive our probability density function. So, Scipy provides us with a class which will allow us to be able to do that. So, the first thing we want to do is start by importing our libraries. And then the first thing we do is we'll create our t-distribution. And our t-distribution, we want to be 25 degrees of freedom. So, we're going to call the Python class, and use the 25 to indicate the degrees of freedom that we want to be able to use.

Next, we're going to calculate the values, which is going to use the line space function which gives us the range of the function, which is between minus 4 and positive 4, which we had expected it to be, which we know it will be actually. And then we're going to create some labels, plot the probability distribution function here with our t-values. So, that's actually what we're doing here. So, the t-distribution function is going to be called with our value

range. And then we're just going to go ahead and show this. So, let's do plot dot show again. And we're going to take a look at this, and this is our probability density function for our t-distribution with degrees of freedom of 25. So, you're going to notice it's very uniform in it's display. So, then we can use the t-distribution object or t-dist, which will give us the cumulative distribution function. So, let's go ahead and plot that as well. And we're still in the beginning stages where we're just sort of looking at what the functions look like.

So next, we're just going to use the same function so here we're taking the t-dist, but we're going to this time use the CDF function, which is from our Python library, and we're going to pass in the same set of values. So, we're going to get a slightly different graph. And if we go ahead and do this, we'll just do a plot dot show again, we can see that here, we get what we expected, which was the cumulative distribution function, for the values with degrees of freedom equal to 25. So, that's great. But if we want to actually run some actual calculations on particular values of x (if you remember, that was the actual value), we want to take a look at both of these situations.

So, in the case of the PDF, if you remember, in the PDF, that will give us the probability for a given value of X, or equal to a given value of x. But the CDF is going to give us the cumulative probability of values less than or equal to x. So, let's take a look at what that would look like. And I wanted to use a graphical representation of this. So, all this looks like a lot of code. I'll explain it to you so that we can get a feel for how we would take a look at this. So, the first thing we want to do is we want to calculate our PDF and CDF values. And if you remember correctly, we kind of already did this. So, we're creating a PDF with a set of t-values that we used before. So, these are just giving us some of the basic values that we're going to need for the functions themselves. Then we create some colors. But here's where it gets a little bit more important in that what we want to do is we want to be able to then select the actual values that we want to calculate the probability distribution function for.

So here, we're taking the t-distribution probability density function for a particular value of x. And so here I'll just enter in the value of x. And then, likewise, we want to do the same thing for the cumulative distribution function for a particular value of x. And then, at the end of it all, we're just going to go ahead, and print everything or display everything that's available. So, let's just go ahead, and run that. Pretty easy to read through the code, but it's allowing us to be able to take advantage of some of the convenience functions that are available in Matplotlib to be able to look at all of the diagrams at once. So, here really just sort of exemplifies the functions that we were just talking about. So, from a cumulative distribution function, I want to be able to look for where my x is negative 1.5 and look at the area that would be under that curve would be 0.07 is my probability. And then likewise, if I were to look at the value where it was equal to x at minus 1.5, I could see the actual value that I would get from there as well. And we'll talk about that in a second when I actually run the functions.

So, here is a really convenient way to be able to take a look at what this graphically would look like if you compare PDF and CDF functions together. And remember, the most important part of this is that the PDF gives you the actual probability for a value when it's equal to x. And then cumulative distribution function would give you the value for the cumulative sum of the probabilities for values less than or equal to x. And so, here I just went through the calculations, and we used the print statement to be able to show them to us. So, here we're looking at the first we're looking at the CDF, where the value is going to

be less than or equal to the value of 1.5. And then secondly, just compare that to the PDF, where we're looking at the value equal to 1.5. And we went through the calculations earlier. So, let's just use the convenience functions. Then here we can see that in the case of the CDF, the fact that 1.5, the cumulative distribution, is about 92%. And then the PDF, meaning that the value actually equal to 1.5, is about 12%.

So, in wrapping up, we talked about what the cumulative distribution function is. We went through a number of examples for determining the P values from the CDF. Hope you enjoyed this conversation and thank you.

****

----------**END**----------