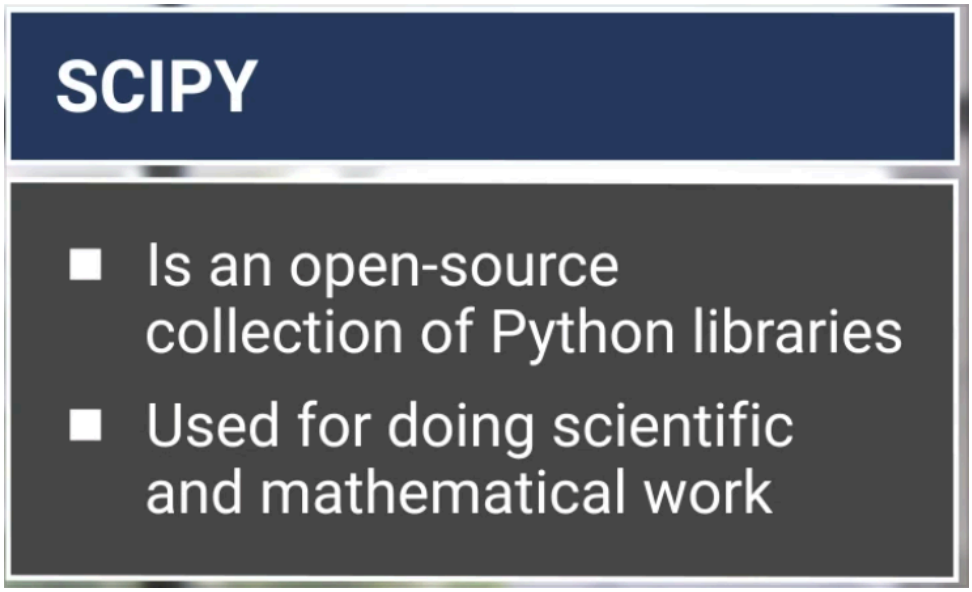
Module 2

Fundamentals of Machine Learning

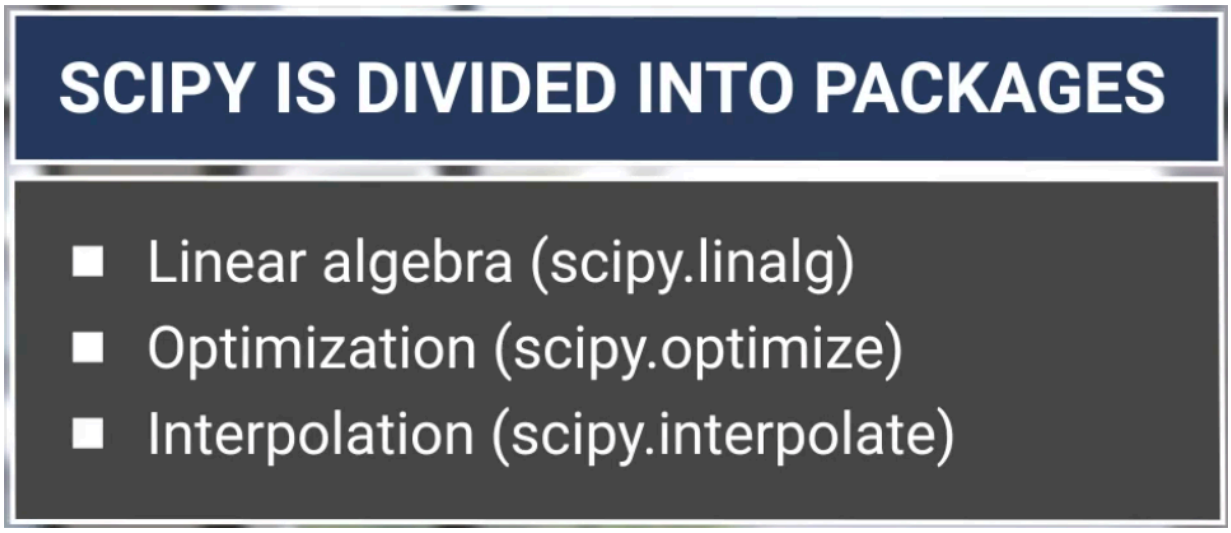
Video 1: Uniform Distribution

Today we will talk about two very common distributions: the uniform distribution, and the normal distribution. And we will begin to work with them in code. Distributions are a basic building block of statistics. Scientists and statisticians over the past couple centuries have discovered many interesting distributions from modeling a wide variety of real-world phenomena, from patterns of wind speeds to human lifespans, to asset returns for investment portfolios.

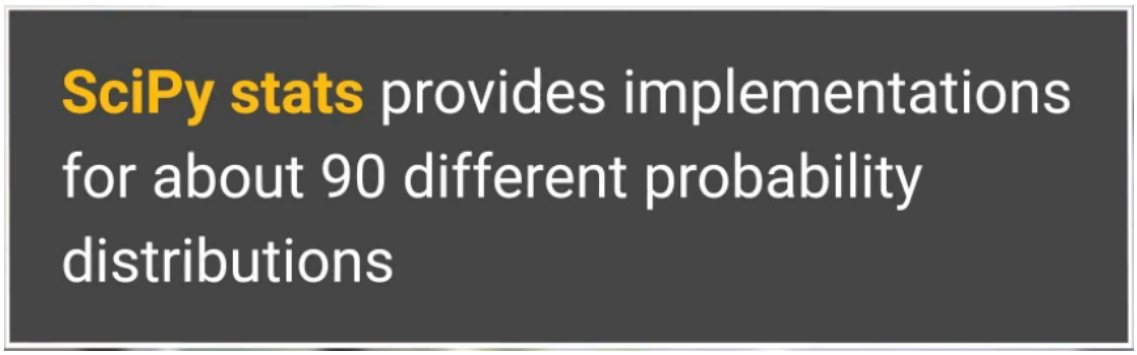
SciPy is an open-source collection of Python libraries for doing scientific and mathematical work.



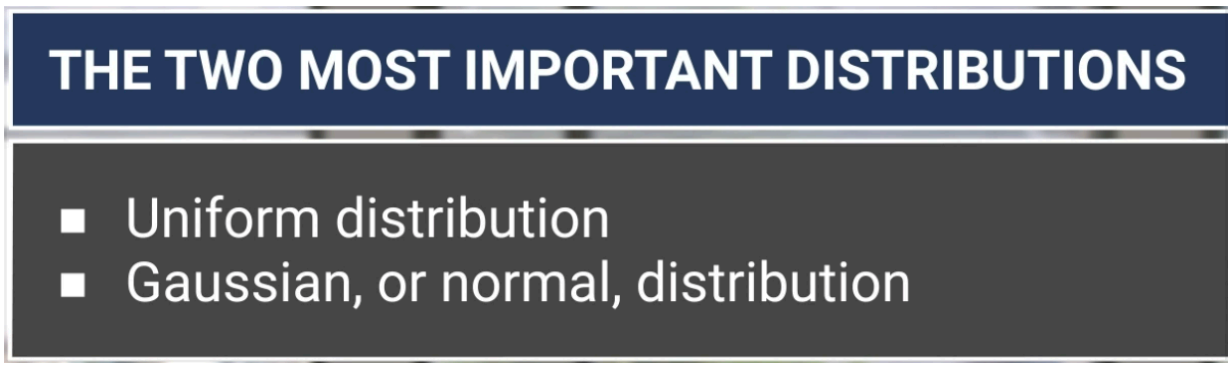
SciPy is divided into a number of packages covering different areas, such as linear algebra, optimization, interpolation, and others.



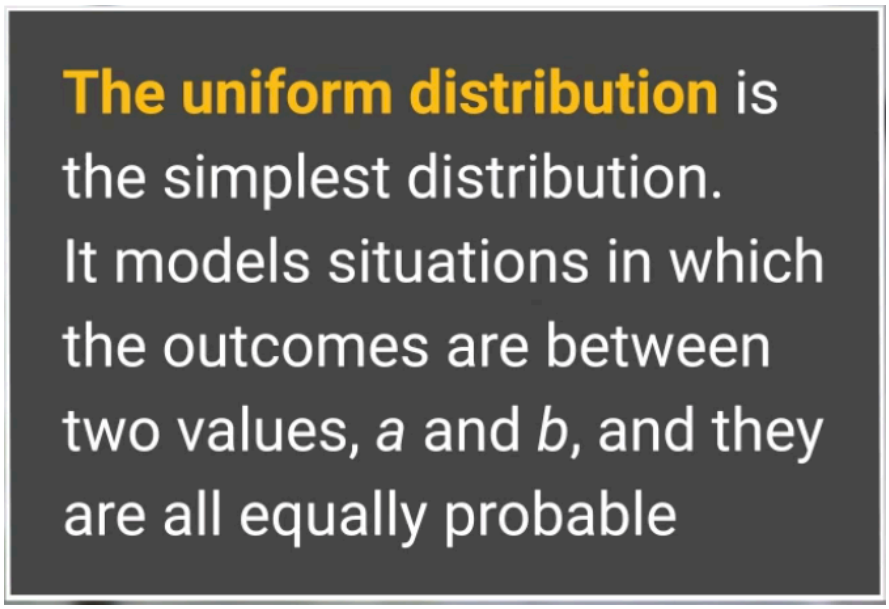
We will be working with the statistics package of SciPy called SciPy stats. SciPy stats provides implementations of about 90 different probability distributions.



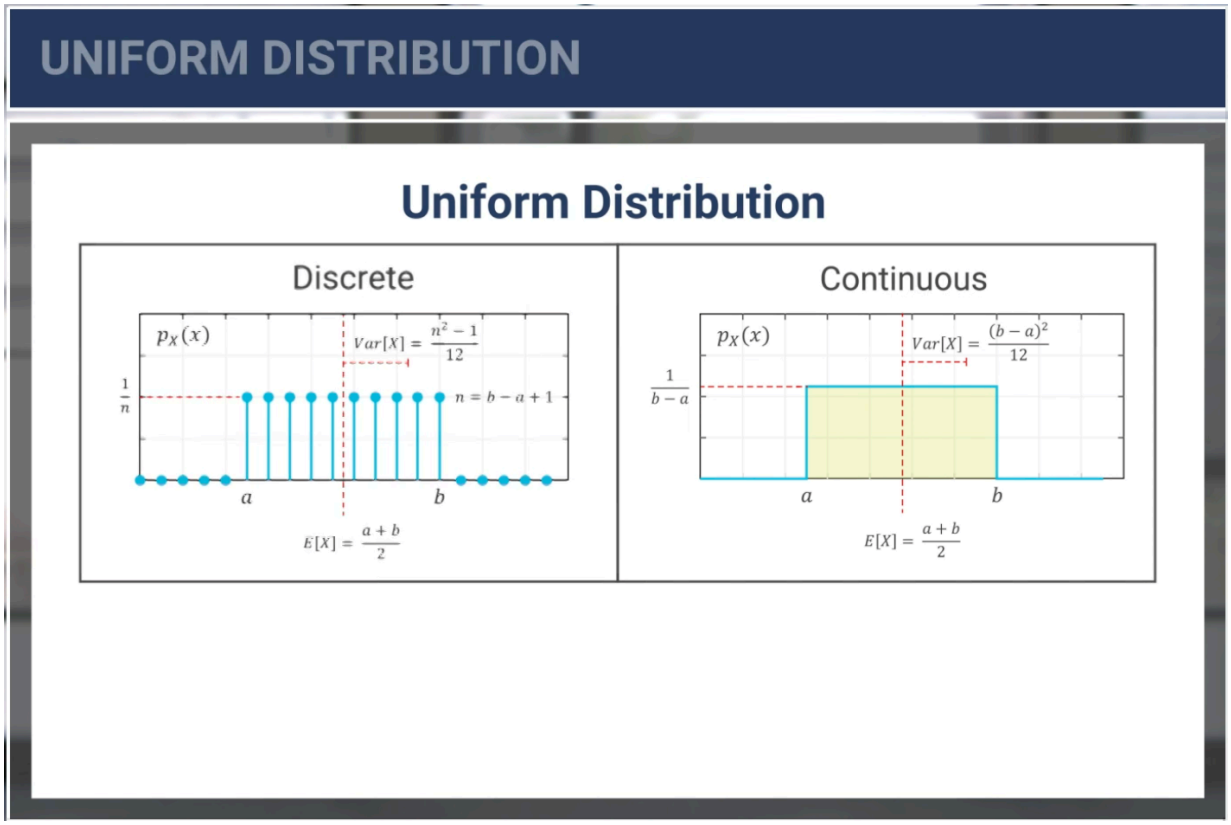
In this lecture, we will introduce the two that are most important for data scientists: the uniform distribution, and the Gaussian or normal distribution.



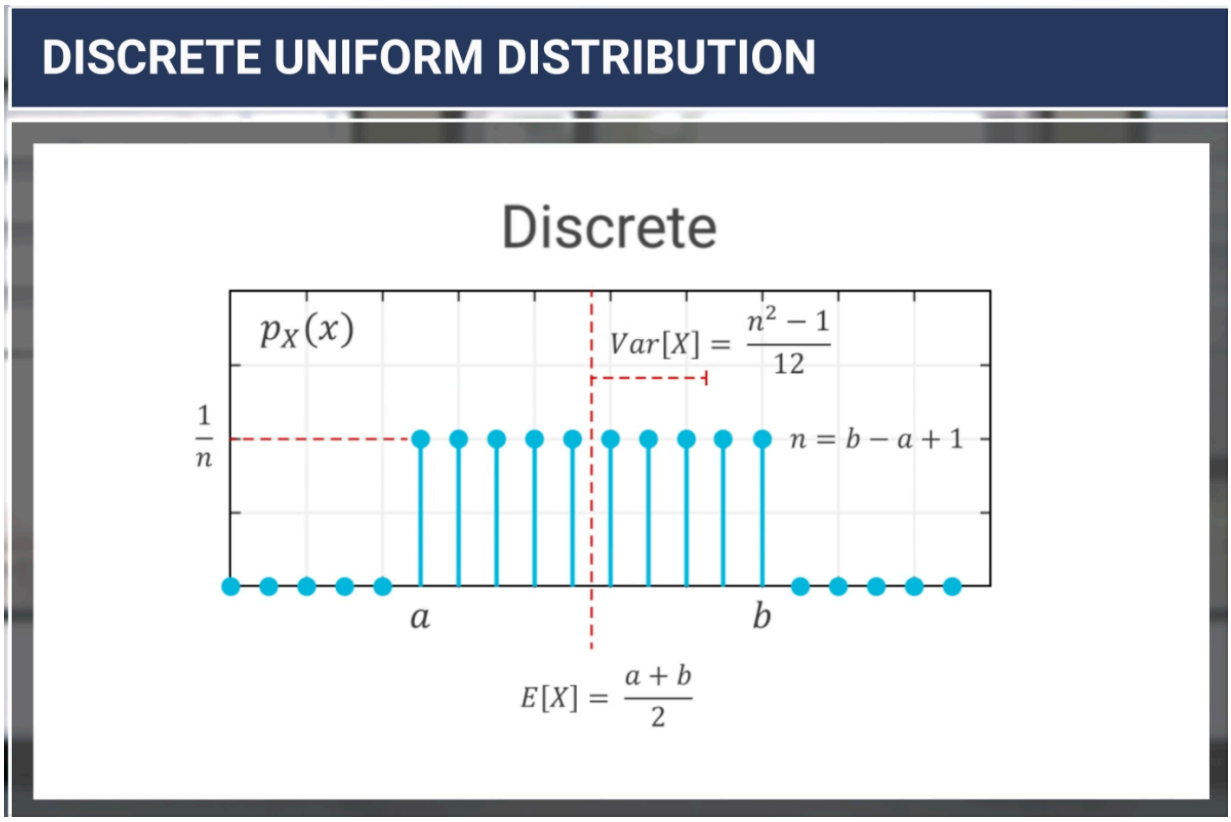
The uniform distribution is the simplest distribution. It models situations where the outcomes are between two values, a and b, and they are all equally probable.



You can see here, the two versions of the uniform distribution: discrete and continuous.



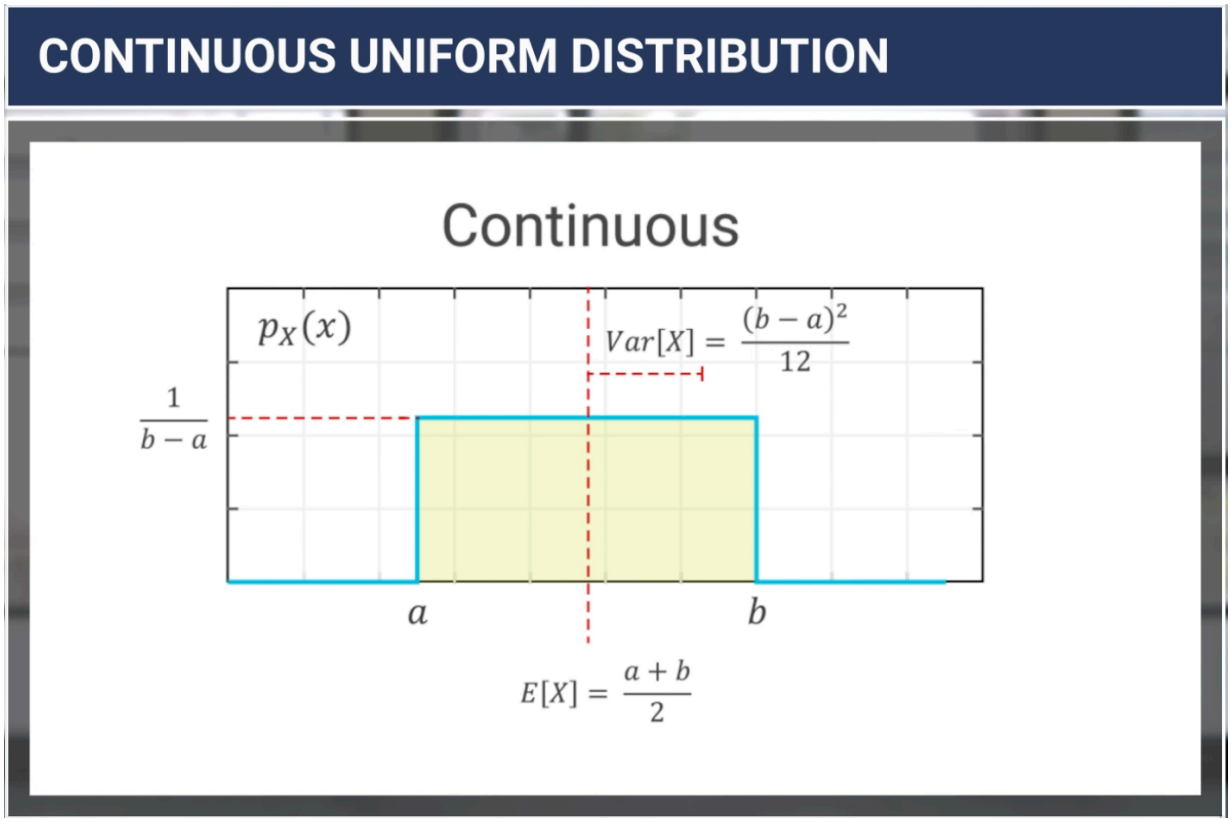
In a discrete uniform distribution, there are n possible outcomes ranging from a low value a to a high value b.



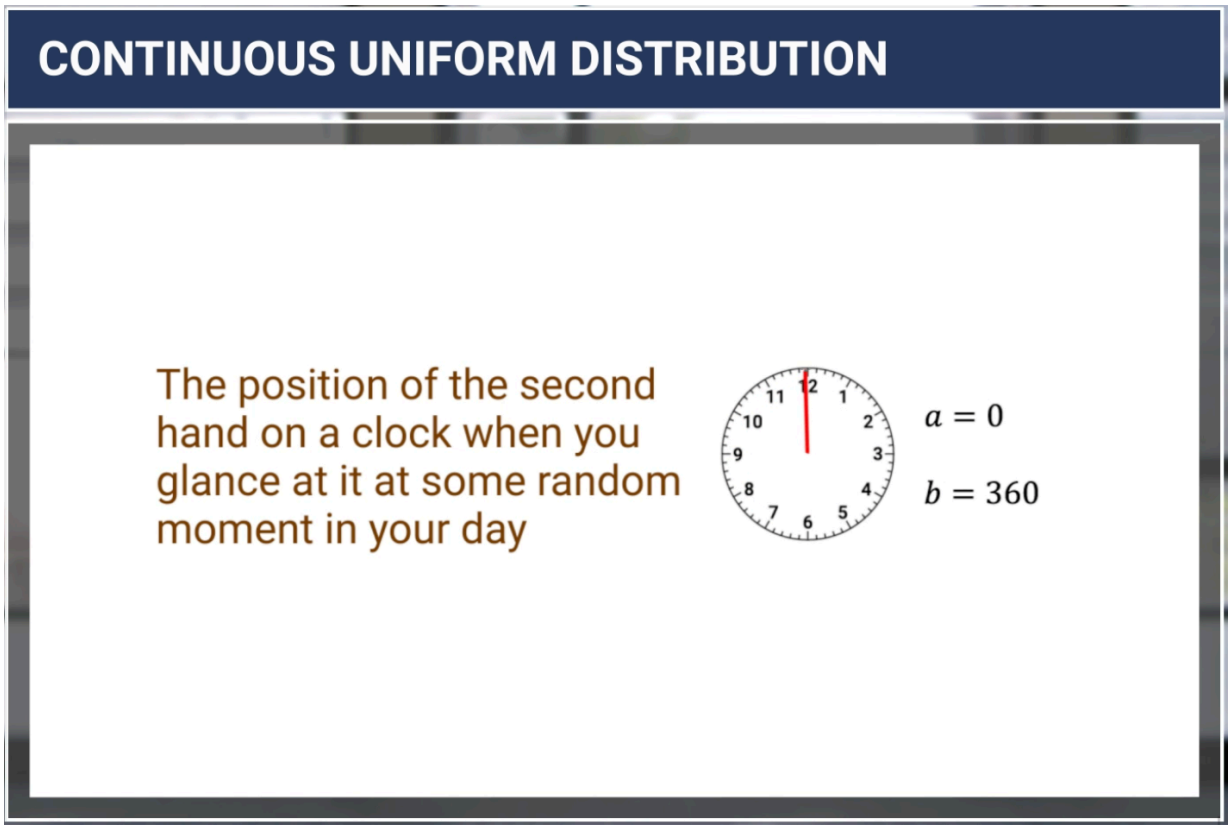
For a single roll of a die, the low value is one and the high value is six, and there are a total of n = 6 possible outcomes.

The probability of any one outcome, is 1/n.

In a continuous uniform distribution, all values between a and b are possible. The probability density function equals 1/(b – a), so that the area under the curve, which is the base times the height, equals one.



An example of a continuous uniform distribution is the position of the second hand of a clock when you glance at it at some random moment in your day. The angle is confined to be between zero and 360 degrees and all values within that range are equally likely.



The expectation of both discrete and continuous uniform distributions is at the midpoint between a and b. Not surprisingly, because both of these are symmetric distributions. The formulas for the variance, as you can see, are a little different. The variance of a discrete distribution is (n2 – 1)/12, while for the continuous distribution, it is (b – a)2/12.

The Variance of a Discrete Distribution

$$Var[X] = \frac{n^2 - 1}{12}$$

The Variance of a Continuous Distribution

$$Var[X] = \frac{(b - a)^2}{12}$$

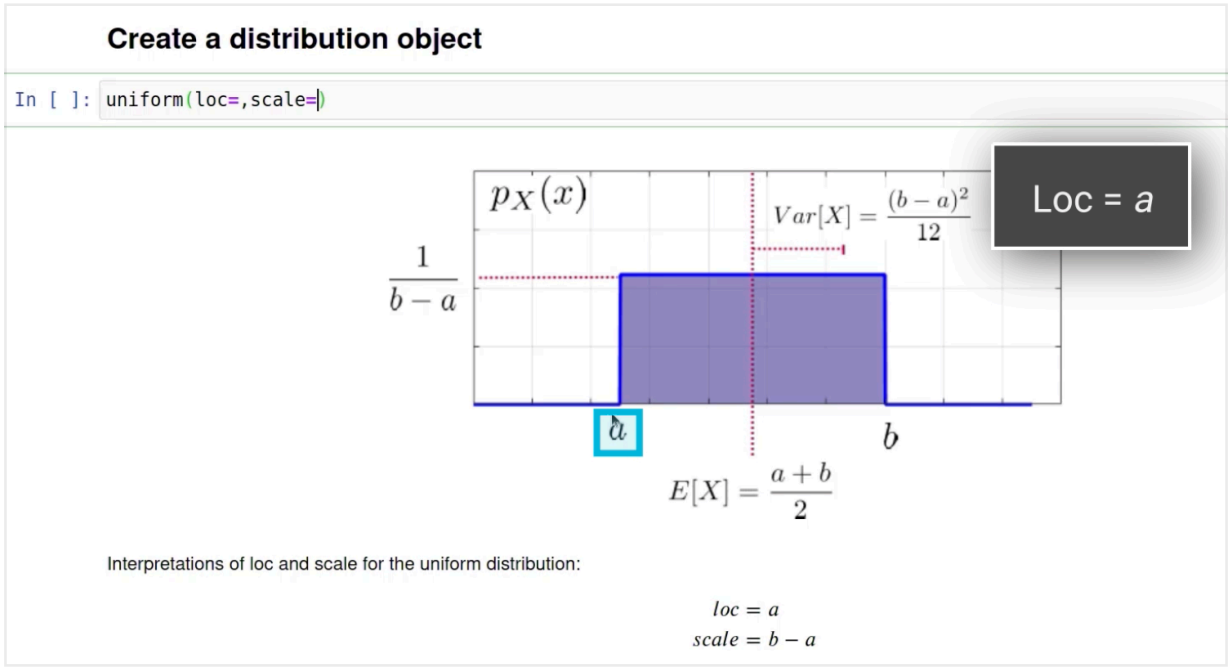
Let's jump to a Jupyter notebook to see how we can create and sample uniform distributions in code.

Video 2: Uniform Python

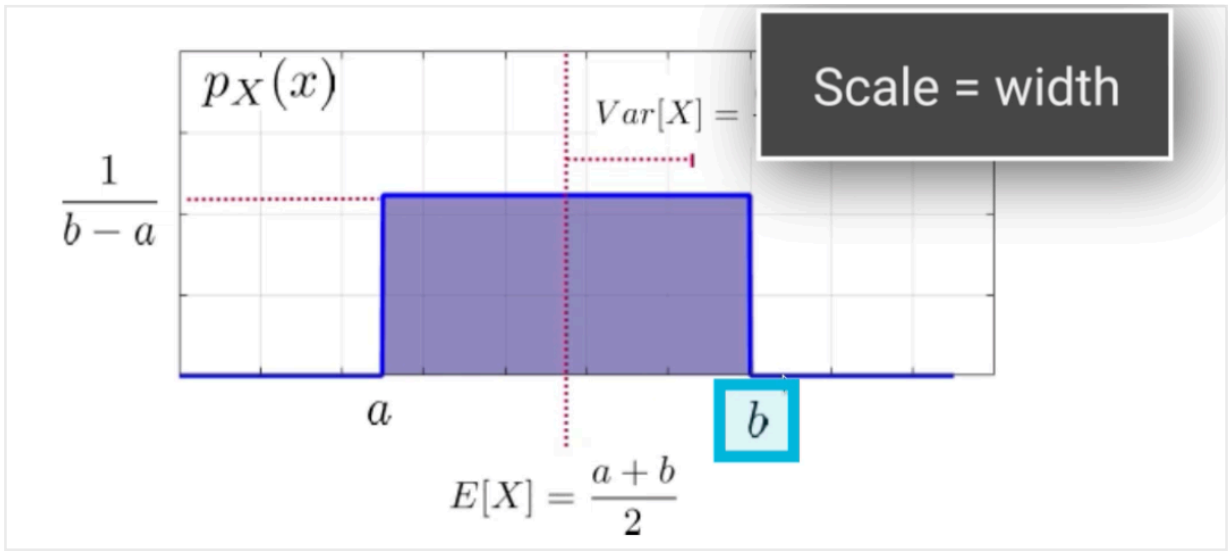
In this video, we will start using SciPy stats for working with probability distributions. Let's look at the documentation. This is the documentation for SciPy stats, and you can see here that it includes discrete and continuous distributions. If I click on Discrete (Statistical) Distributions and scroll down to the bottom, I get the list of all of the discrete distributions that it supports. And in the Continuous (Statistical) Distributions setting, I get a whole bunch of continuous distributions.

For now, we are going to look at the Uniform Distribution, which is here. And the implementation has the real documentation for uniform distribution. And you can see a bunch of examples, and the list of attributes and methods. I have linked that here, under Continuous uniform distributions. And it also tells us what is the proper import statement for this class. So we put that here to import the uniform distribution class, as well as our standard imports NumPy and Matplotlib.

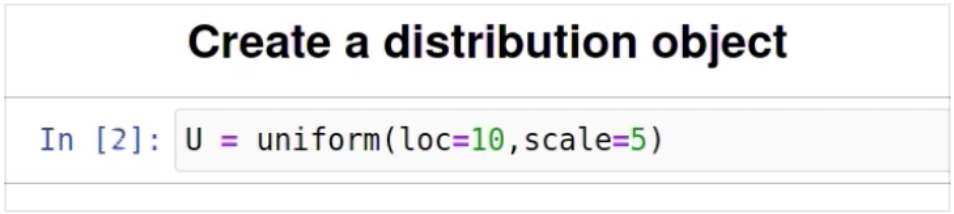
So how do we use this? First, we have to create a distribution object. And you'll recall from the lecture that a uniform distribution looks like this. It spans from a to b, and its value is 1/(b – a). So to create the uniform distribution object, we now just call it uniform, and we have to pass it the parameters. And the parameters are going to be called loc and scale. These are common parameters for all of the distributions in SciPy stats, and they take on different meanings depending on the distribution that you're using. For a uniform distribution, loc is the left edge of the distribution,



and scale is the width.

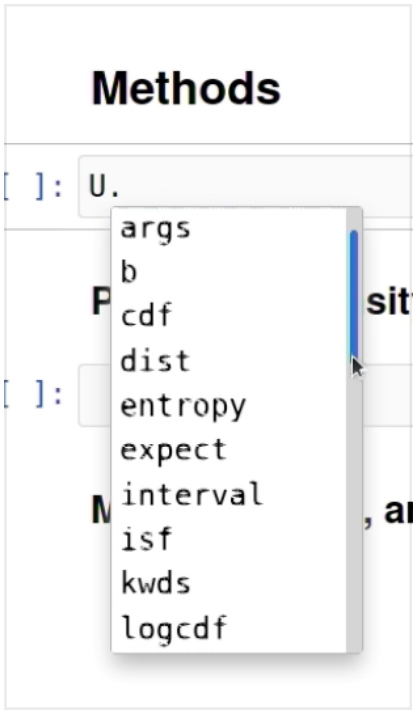


So for example, if we wanted to create a distribution that went from 10 to 15, we would say loc equals 10 and scale, which is the width, would equal to 5. So let's create that uniform distribution and assign it to the variable U.

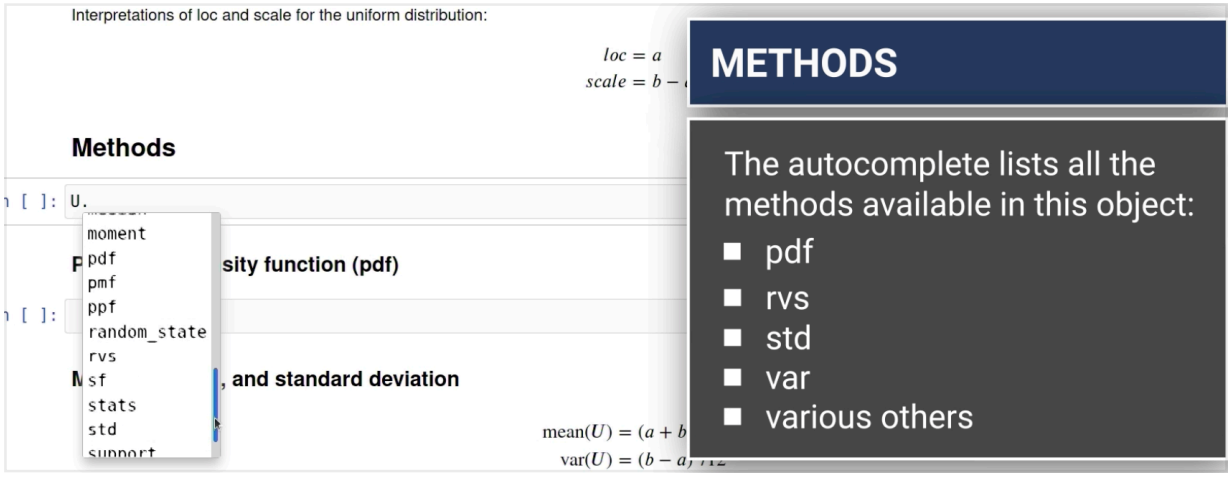


There we have it.

Now, what methods are available to us in this object? What are the methods that we can call for this uniform object? We can see the list by typing U dot ("U."), and then the Tab key, which gives us the autocomplete and lists all of the methods that are available in this object.



We can see here, the pdf function, the rvs function, which is the sampling function, the standard deviation, the variance, and various others.



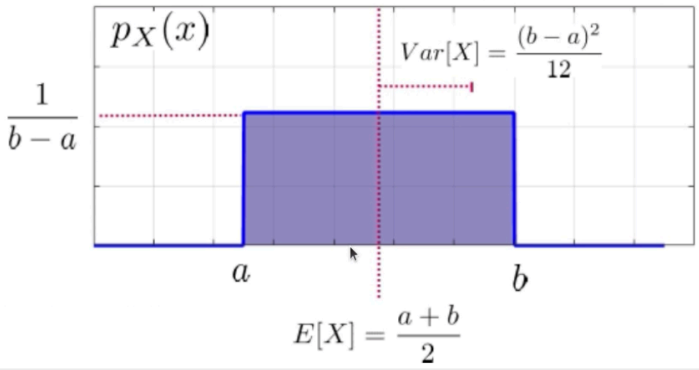
So we'll take a look at a few of these. First of all, the pdf function.

Let's call u.pdf at 8, and this is 0. That's pretty reasonable because 8 is to the left of 10, so the value there should be 0.

Now, if I call it at, say 12, I'll get 0.2, which is what we'd expect as well.

Create a distribution object

```
In [2]: U = uniform(loc=10,scale=5)
```



We can call pdf on an array. So let's say 8, 12, and say, 20.

Probability density function (pdf)

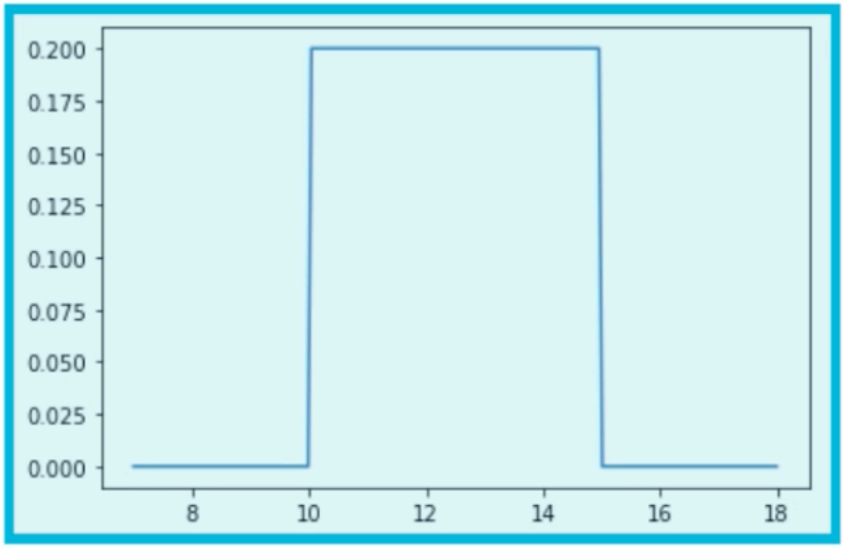
```
In [15]: U.pdf([8,12,20])
Out[15]: array([0. , 0.2, 0. ])
```

And get 0, 0.2, and 0. Let's do this. Let's create a large array of evenly spaced values between 7 and 18. And we're going to use NumPy's linspace function for doing this. So we're going to create values between 7 and 18, and we're going to create 200 of them. So let's call that upoints. And then we're going to pass upoints into the pdf. And we'll call that ppoints. What comes out. And so, now we can plot upoints versus ppoints. This is what we obtain. It's a picture of the pdf.

Probability density function (pdf)

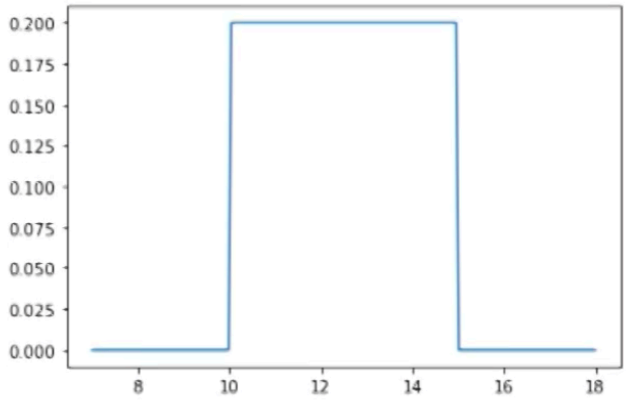
In [16]:
```
upoints = np.linspace(7,18,200)

ppoints = U.pdf(upoints)

plt.plot(upoints,ppoints)
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x7feb23cabdf0>]
```



OK. Well, another thing that we can do is evaluate the mean, the variance, and the standard deviation of the uniform distribution. And from theory, we already know what the values of that should be. So if I say the mean, that is the center value, which is between 10 and 15. It should be 12.5.

Out[16]: [<matplotlib.lines.Line2D at 0x7feb23cabdf0>]

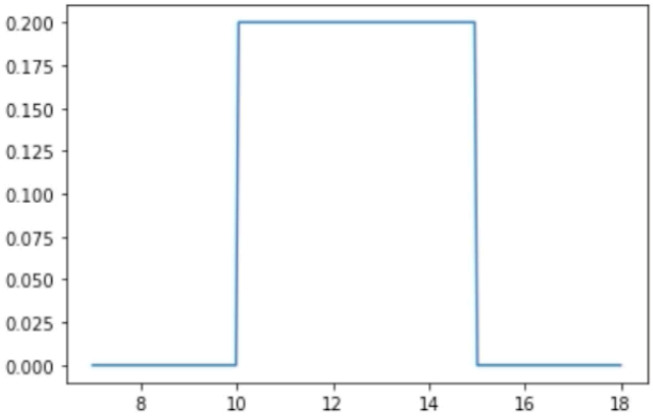


Mean, variance, and standard deviation

$$\text{mean}(U) = (a + b)/2$$
$$\text{var}(U) = (b - a)^2/12$$
$$\text{std}(U) = (b - a)/\sqrt{12}$$

```
In [17]: U.mean()
Out[17]: 12.5
```

And that's indeed what we get. The variance is b minus a (which is 5) squared, which is 25, divided by 12, will be a little bit more than 2, OK.

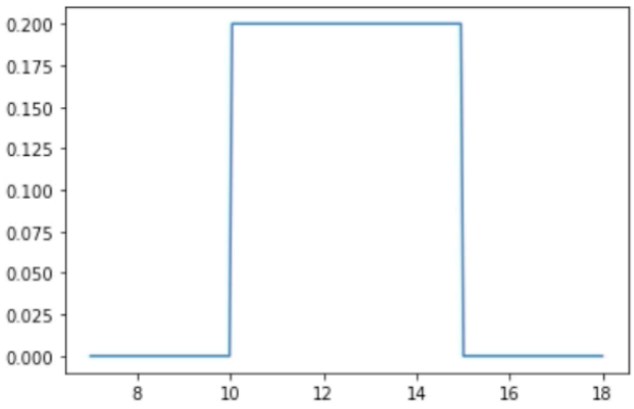


Mean, variance, and standard deviation

$$\text{mean}(U) = (a+b)/2$$
$$\text{var}(U) = (b-a)^2/12$$
$$\text{std}(U) = (b-a)/\sqrt{12}$$

```
In [18]: U.var()
Out[18]: 2.083333333333333
```

And the standard deviation is the square root of that. So the square root of 2 is about 1.4. OK, there you have it.



Mean, variance, and standard deviation

$$\text{mean}(U) = (a+b)/2$$
$$\text{var}(U) = (b-a)^2/12$$
$$\text{std}(U) = (b-a)/\sqrt{12}$$

```
In [19]: U.std()
Out[19]: 1.4433756729740643
```

So those are the values of the mean, variance and standard deviation.

Next, the other important method that we are going to use is rvs, which stands for random variates. And basically what that does is that it samples the distribution. So if I call that, I get a single sample of this uniform distribution.

Sampling (rvs)

```
In [20]: U.rvs()
Out[20]: 10.120673849173219
```

And I can call it many times. I'm going to press Ctrl+Enter here, many times, and you see the value changing.

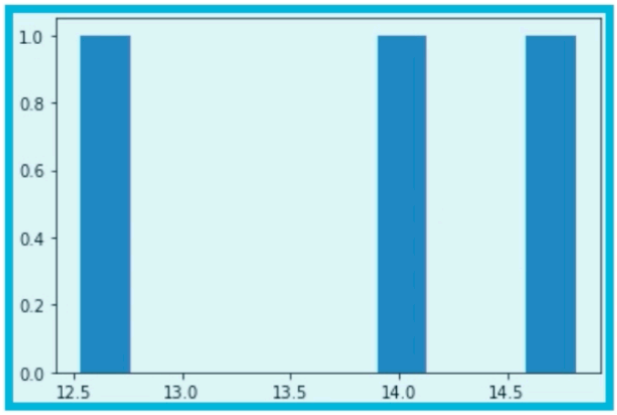
```
Sampling (rvs)

In [34]: U.rvs()
Out[34]: 13.159319331405515
```

We get numbers all between 10 and 15. OK. So, we can sample many times at once by passing in a size parameter. So if I say size=3, I'm going to get an array of three samples.

```
Sampling (rvs)

In [35]: U.rvs(size=3)
Out[35]: array([13.60651855, 11.75172872, 13.64383427])
```

OK. This is useful. We can make a histogram with all of these samples. So I'm going to put this into a histogram. And now we have a histogram with three samples, which doesn't look very uniform.

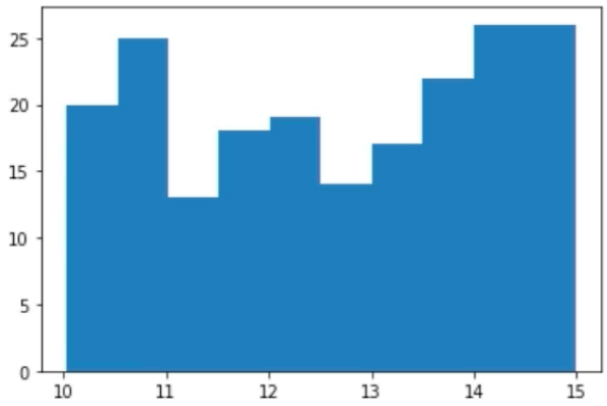
```
In [36]: plt.hist(U.rvs(size=3))
Out[36]: (array([1., 0., 0., 0., 0., 0., 1., 0., 0., 1.]),
          array([12.53412633, 12.76242028, 12.99071424, 13.2190082 , 13.44730215,
                 13.67559611, 13.90389007, 14.13218402, 14.36047798, 14.58877193,
                 14.81706589]),
          <BarContainer object of 10 artists>)
```



We can make it, say, 100 samples, or 200 samples. And let's compare this to the

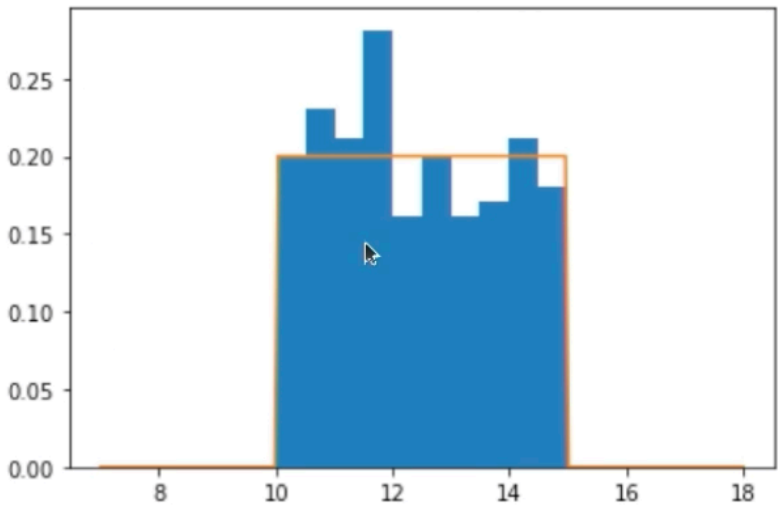
plot that we had before, to this plot.

```
In [38]: plt.hist(U.rvs(size=200))

Out[38]: (array([20., 25., 13., 18., 19., 14., 17., 22., 26., 26.]),
          array([10.030802  , 10.52634293, 11.02188387, 11.51742481, 12.01296574,
                 12.50850668, 13.00404761, 13.49958855, 13.99512949, 14.49067042,
                 14.98621136]),
          <BarContainer object of 10 artists>)
```



Let's put this into the same plot. And now we see that they're in the same range, but the histogram looks a lot taller. So, to scale the histogram to the size of the distribution, we have to say in the histogram, density=True. And then it's going to scale it correctly. So we see that the histogram begins to look a lot like the distribution, the larger we make the size of the sample.

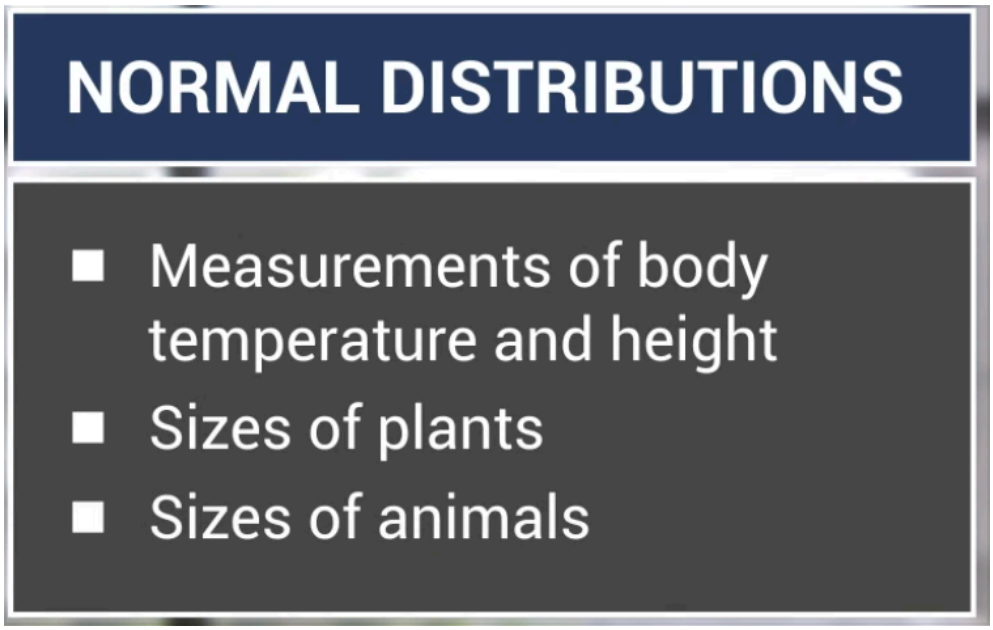
```
In [40]: plt.hist(U.rvs(size=200),density=True)
         plt.plot(upoints,ppoints)

Out[40]: [<matplotlib.lines.Line2D at 0x7feb22e11820>]
```



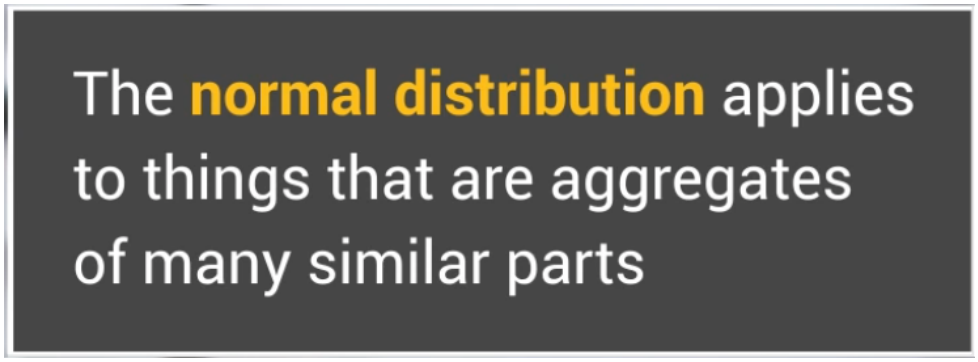
So if I make it 1,000, it's looking more similar, 10,000, even more similar, 100,000, we're basically the same.

Video 3: Gaussian Distribution

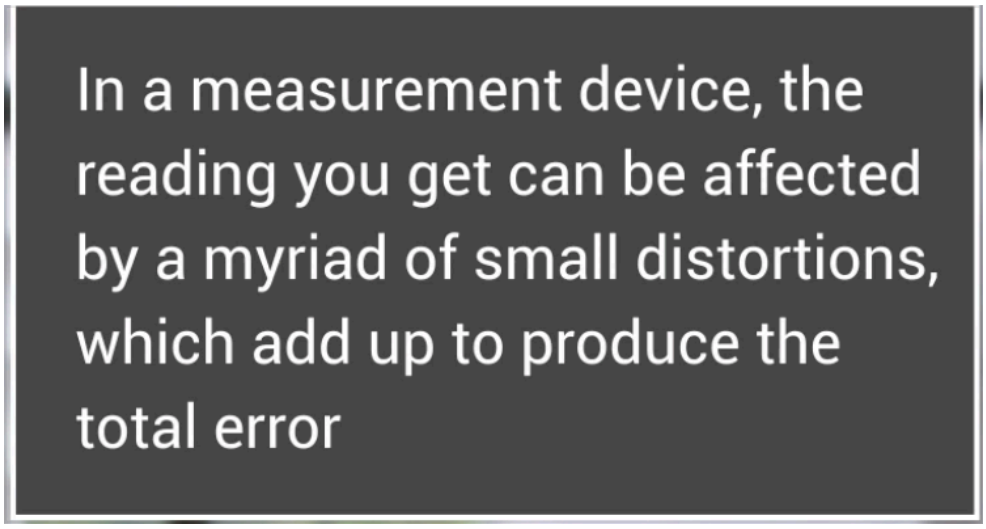
The Gaussian or normal distribution is the most important and widely used distribution for statisticians and scientists. It is often referred to as the bell curve, and it pops up a lot in data science applications. Measurements of things like body temperature, height, sizes of plants and animals, are often normally distributed.



And we also find normal distributions in the world of human technology, in manufacturing, and in economics. So why is the normal distribution so ubiquitous? We will get to that in the next slide, but generally speaking, the normal distribution applies to things that are aggregates of many similar parts.

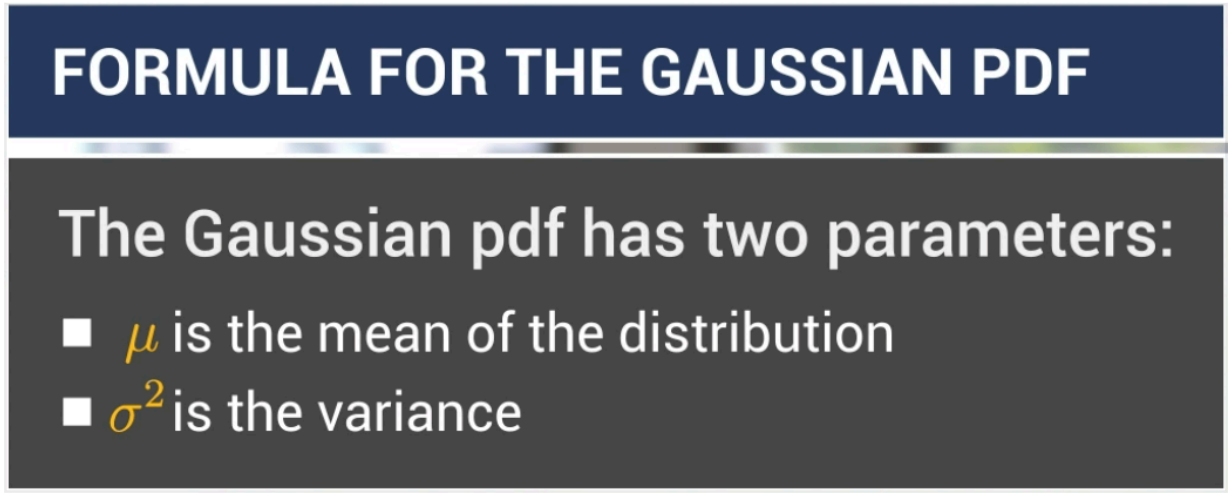


So for example, in a measurement device, the reading you get can be affected by myriad small distortions which add up to produce the total error.

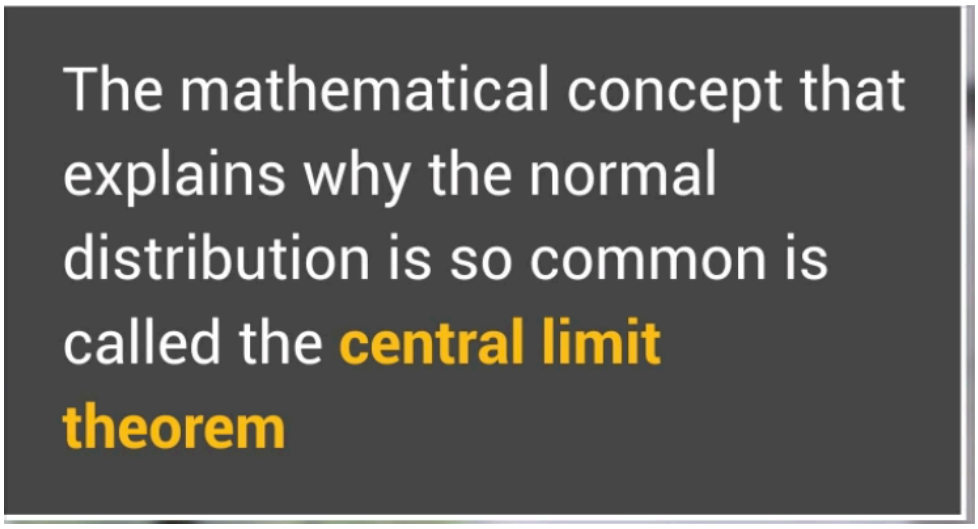


This adding up of small variations is what produces a normal distribution.

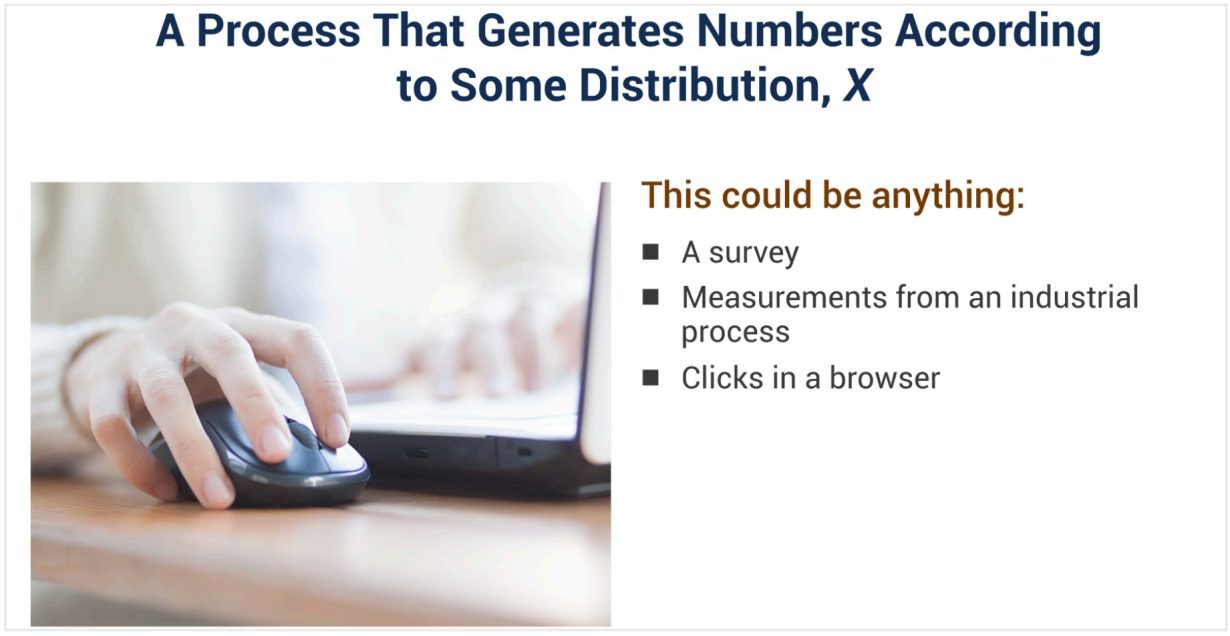
Here is the formula for the Gaussian pdf. It has two parameters. Mu is the mean of the distribution. It is located in the middle. Again, not surprising because the distribution is symmetric. And sigma squared is the variance.



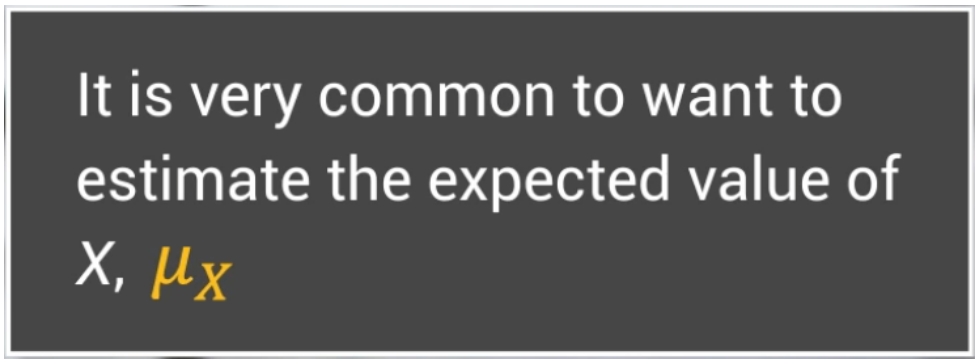
Let's see why the normal distribution is so common. The mathematical concept that explains this is called the central limit theorem. We will not prove the theorem or even state it in mathematical terms here. Our goal is just to understand the idea.



So, consider this experiment. Say you have a process that generates numbers according to some distribution, X. It could be anything, a survey measurements from an industrial process, clicks in a browser, etc.

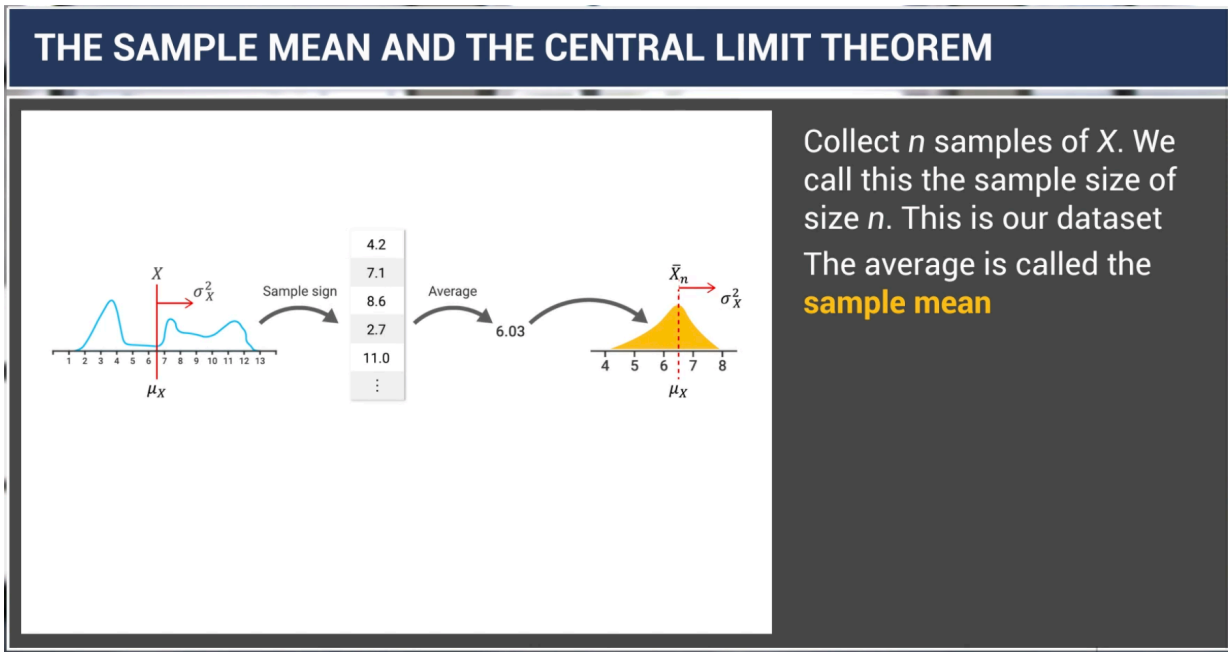


We do not know the distribution of X. All we can see are the samples, the data. A very common thing to want to do is to estimate the expected value of X, mu_X.

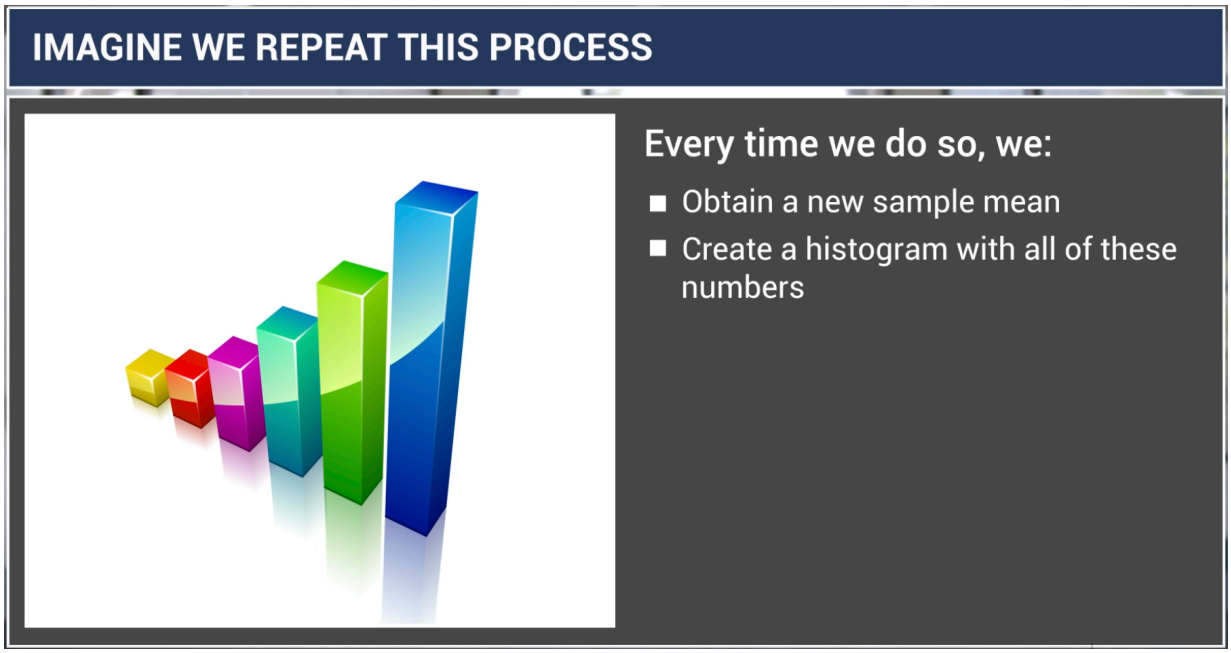


To do this, we collect n samples of X. We call this the sample size of size n. This is our dataset.

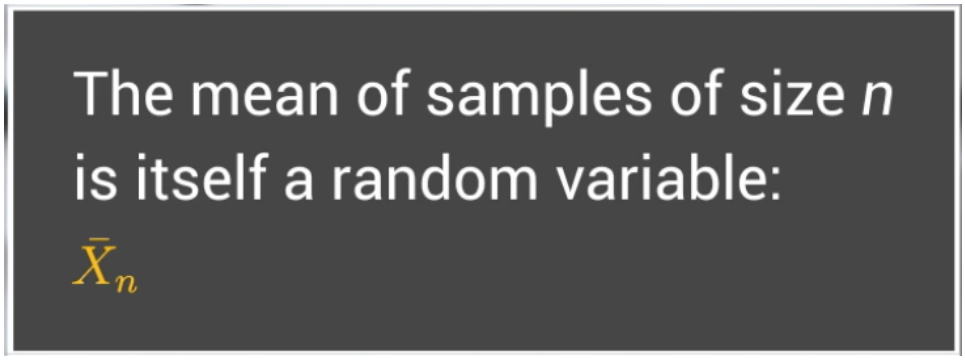
Here is one such sample. It is a list of numbers 4.2, 7.1, and 8.6, etc. A reasonable thing to do is to estimate mu_X, is to take the average of these numbers. Let's say the average is 6.03. This is called the sample mean.



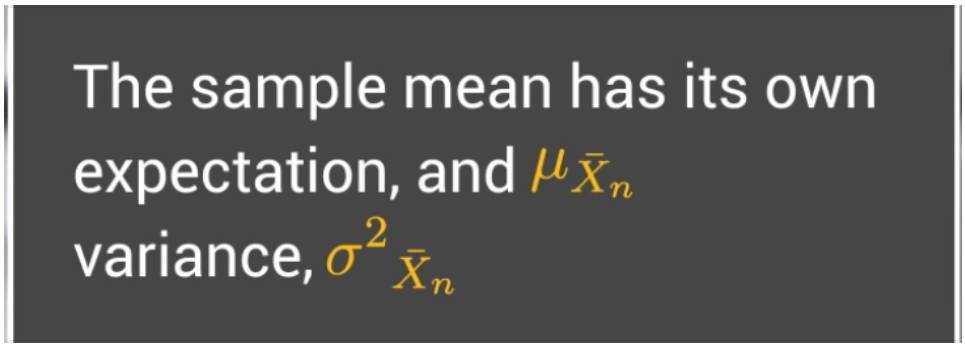
But how good is 6.03 as an estimate of mu_X? Well, to answer this, let's imagine we repeat this process many, many times. Every time we do so, we obtain a new sample mean and we create a histogram with all of these numbers.



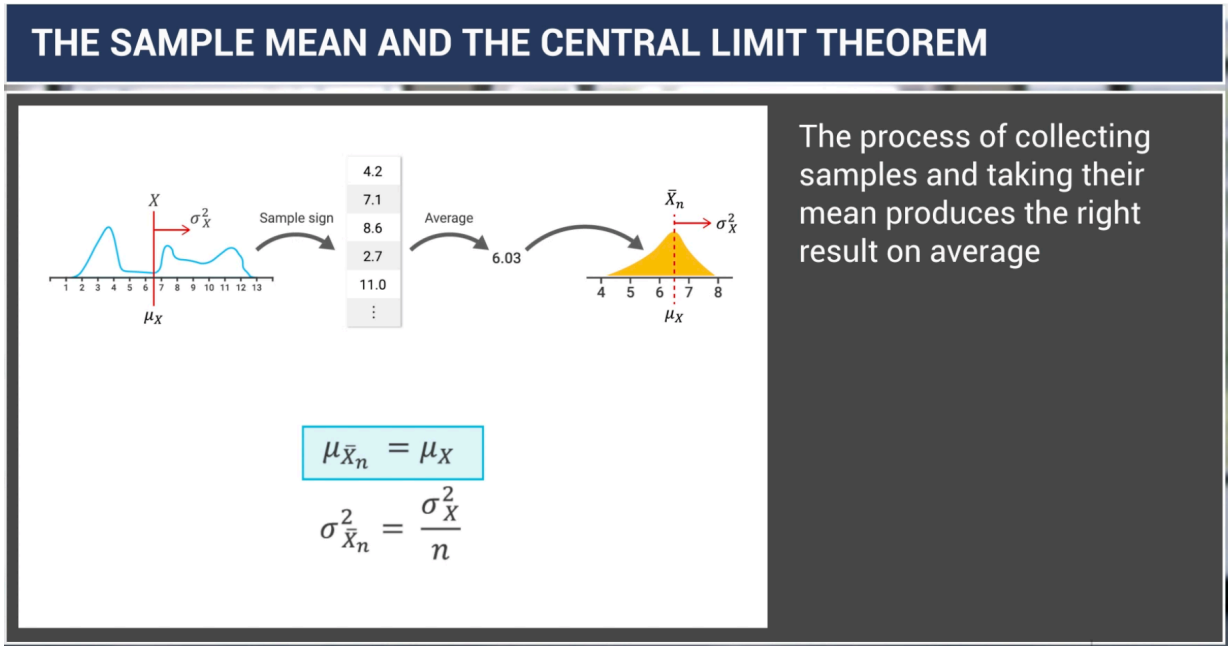
In fact, the mean of samples of size n, is itself a random variable. Let's call this random variable Xbar_n.



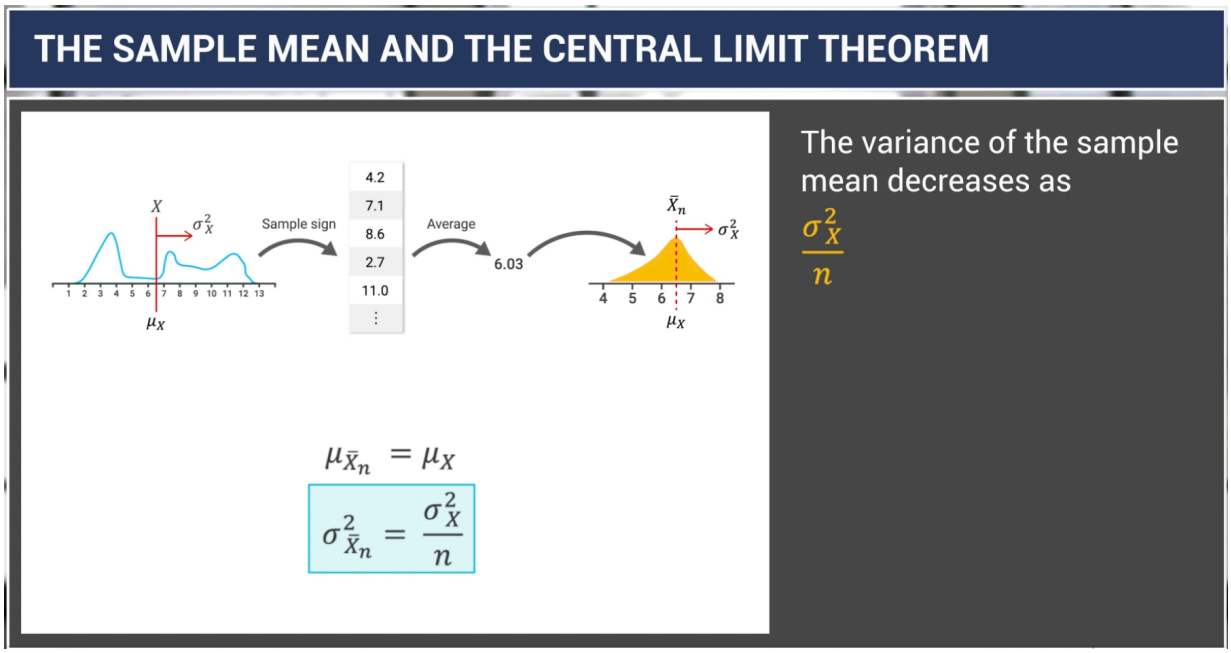
The sample mean has its own expectation, mu_Xbarn, and variance, sigmasquared_Xbarn.

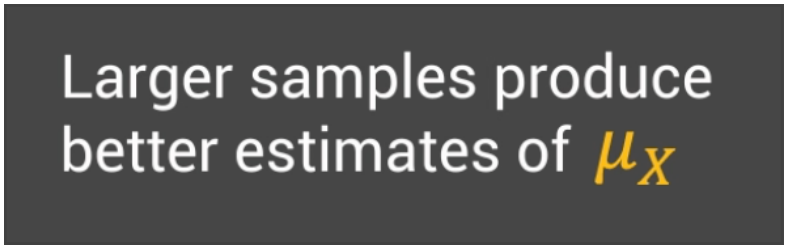


Here are two crucial facts. The first, we know from the law of large numbers. The expectation of the sample mean equals the expectation of X. Although our sample mean may not exactly equal the true mean, that is mu_X is probably not precisely 6.03, at least we can say that the process of collecting samples and taking their mean produces the right result on average.



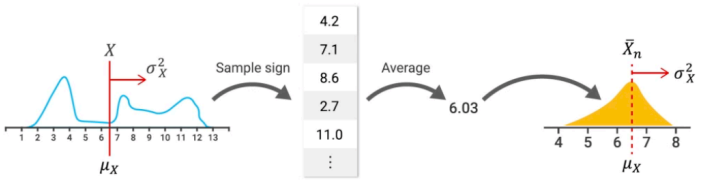
Second, the variance of the sample mean decreases as sigma squared over n. This tells us that the larger the sample, the smaller sigma_Xbarn, and the tighter the sample means will be distributed around the true mean. Larger samples produce better estimates of mu_X.





Now for the central limit theorem. This is a tremendously consequential theorem for statistics and machine learning because it allows us to assume normality in many situations. Whereas the two previous facts told us something about the mean and variance of Xbar_n, the central limit theorem tells us something about its shape. It says that as n increases, Xbar_n becomes closer and closer to a normal distribution.

CENTRAL LIMIT THEOREM



Two important facts:

$$\mu_{\bar{X}_n} = \mu_X$$

$$\sigma^2_{\bar{X}_n} = \frac{\sigma^2_X}{n}$$

The **CLM** tells us something about the shape of $\bar{X}_n$
It says that as *n* increases, $\bar{X}_n$ becomes closer to a **normal distribution**

Notice that these three statements, the two facts and the central limit theorem, apply regardless of the distribution of X.

$$\mu_{\bar{X}_n} = \mu_X \qquad \sigma^2_{\bar{X}_n} = \frac{\sigma^2_X}{n}$$

As n increases, $\bar{X}_n$ becomes $normal$.

The two facts and the CLM apply regardless of the distribution of *X*

X could even be a discrete distribution, it is still true that its sample mean will approach a continuous normal distribution as n grows.

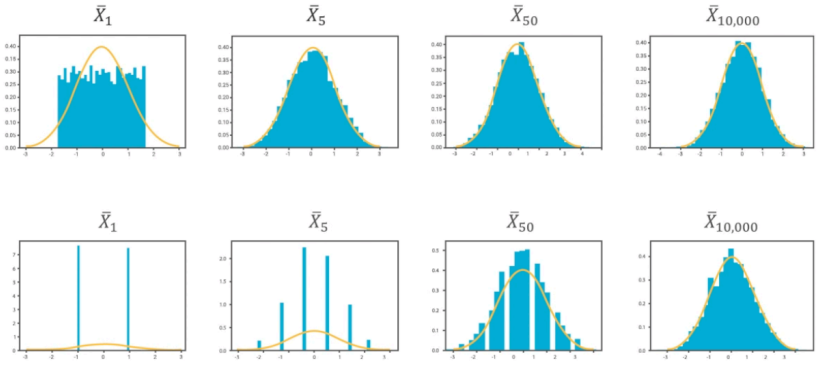
$$\mu_{\bar{X}_n} = \mu_X \qquad \sigma^2_{\bar{X}_n} = \frac{\sigma^2_X}{n}$$

As n increases, $\bar{X}_n$ becomes $normal$.

X could even be a discrete distribution. Its sample mean will approach a continuous normal distribution as *n* grows

Let's see an example. Here we see two progressions of the sample mean, for sample sizes ranging from one to 10,000. For the top plot, the base distribution X is continuous and uniform.

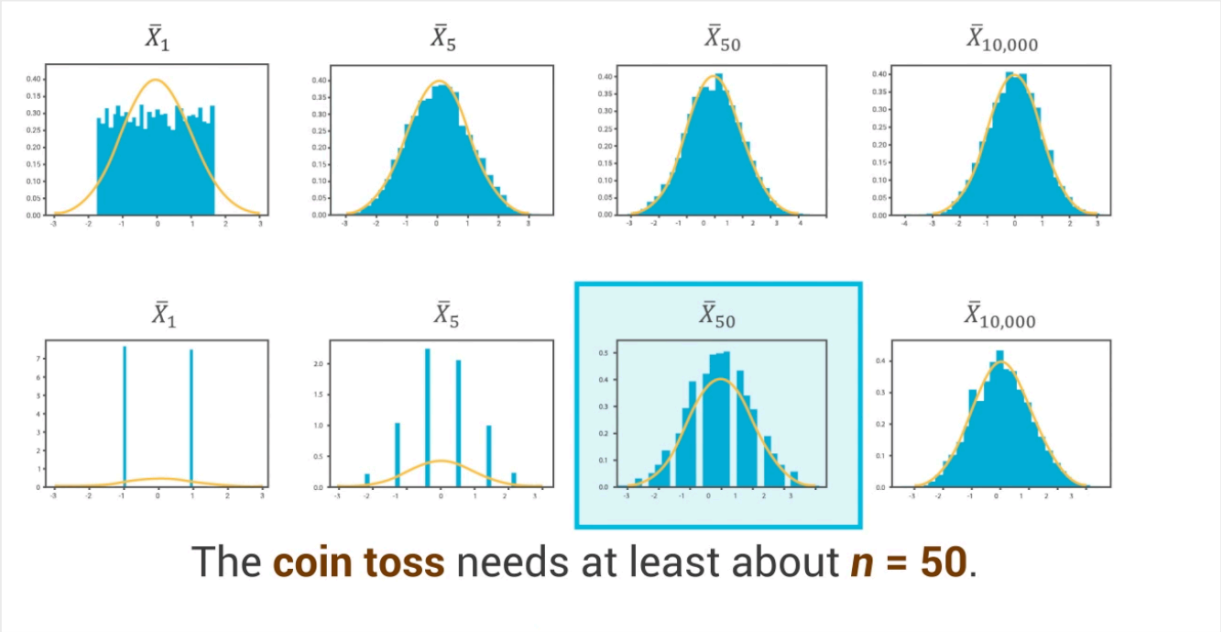
CLM EXAMPLES



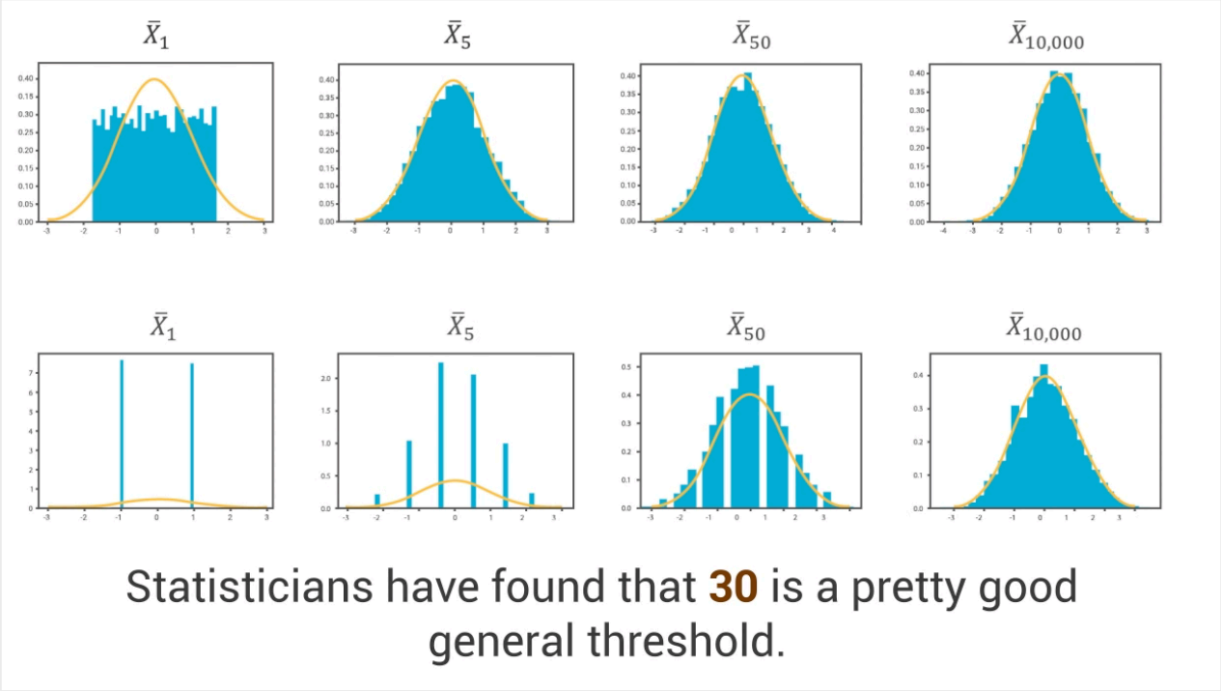
For the bottom, the numbers are generated by tosses of a fair coin, the so-called Bernoulli distribution. The blue bars are a histogram constructed from a large number of sample means. When n=1, the distribution of the sample mean is the same as the distribution of X. And you can see this in the plots. The orange line is a normal distribution. You can see that as n grows from left to right, the sample means for both uniform and Bernoulli distributions approach normality. They adhere more and more closely to the orange line. This is what the central limit theorem predicts.

The central limit theorem does not, however, give us a good threshold for deciding when the normal approximation becomes valid. You can see that for a uniform distribution, the approximation is pretty good at n=5.

But for the coin toss, we need at least about n=50.



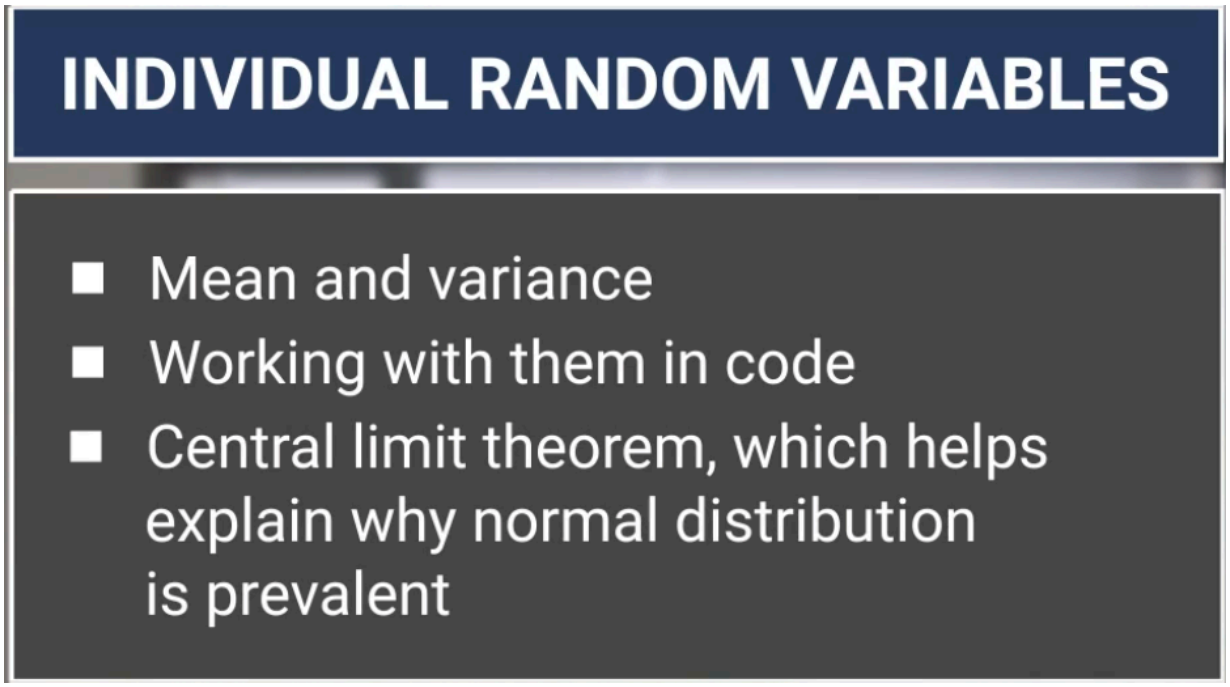
Statisticians have found that 30 is a pretty good general threshold.



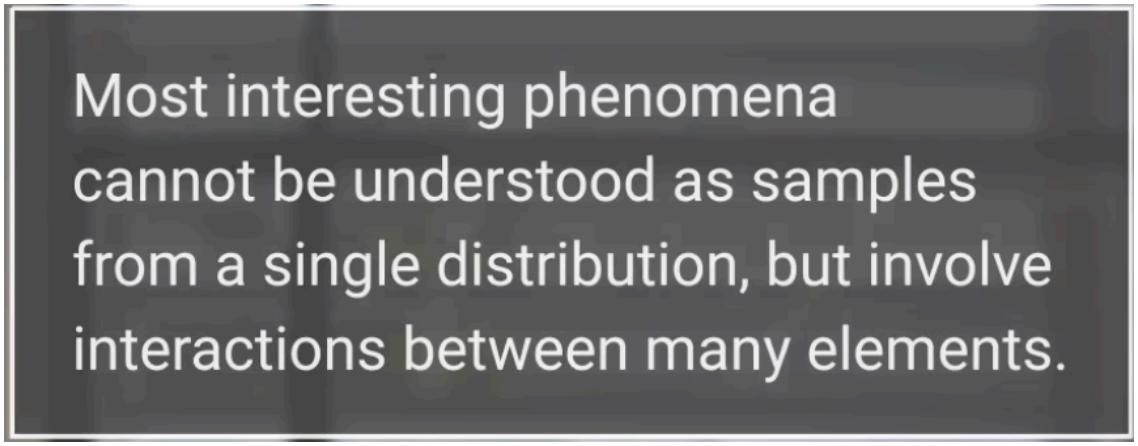
So if you have a sample size of 30 or more, you can assume that the distribution of its mean is normal without having to know much about the process that generated the numbers, beyond that the end samples were taken independently. Thirty is not a huge number, and most of the datasets that we will encounter will be much larger than that.

Video 4: Multivariate Distributions

So far we have studied individual random variables and their distributions and we've learned about their mean and variance, about working with them in code, and also about the interesting fact of the central limit theorem, which helps to explain why the normal distribution is so prevalent in nature.

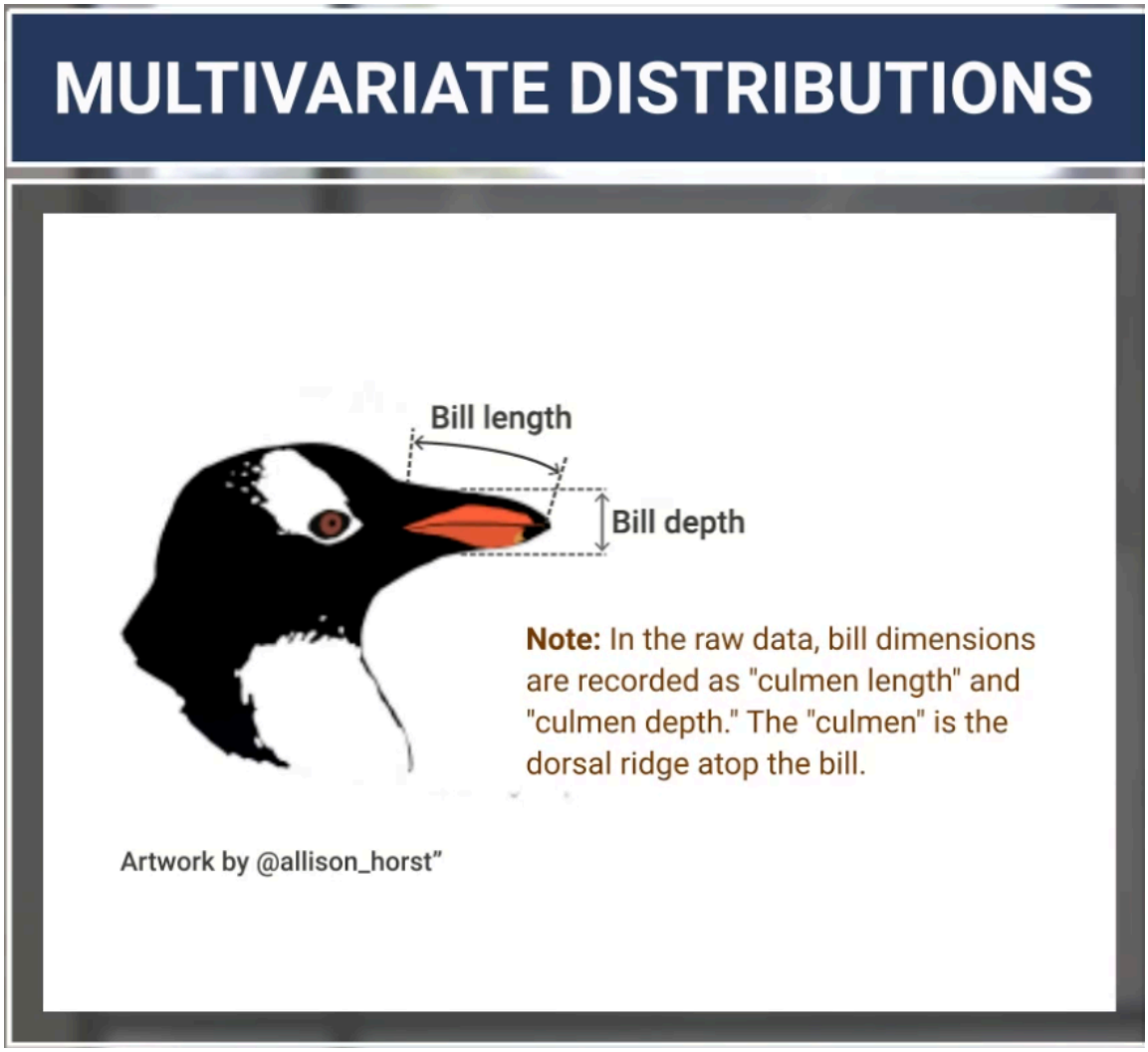


However, the world is more complicated than just a bunch of isolated random variables. Most interesting phenomena cannot be understood as samples from a single distribution, but rather involved interactions between many elements.



Consider a study of Gentoo penguins. Gentoos are these adorable penguins with a white dash above the eyes that live in large numbers in the Falkland Islands and in parts of Antarctica.

A study by Dr. Kristen Gorman of the University of Alaska Fairbanks, measured the length and depth of the beaks of 123 Gentoos.

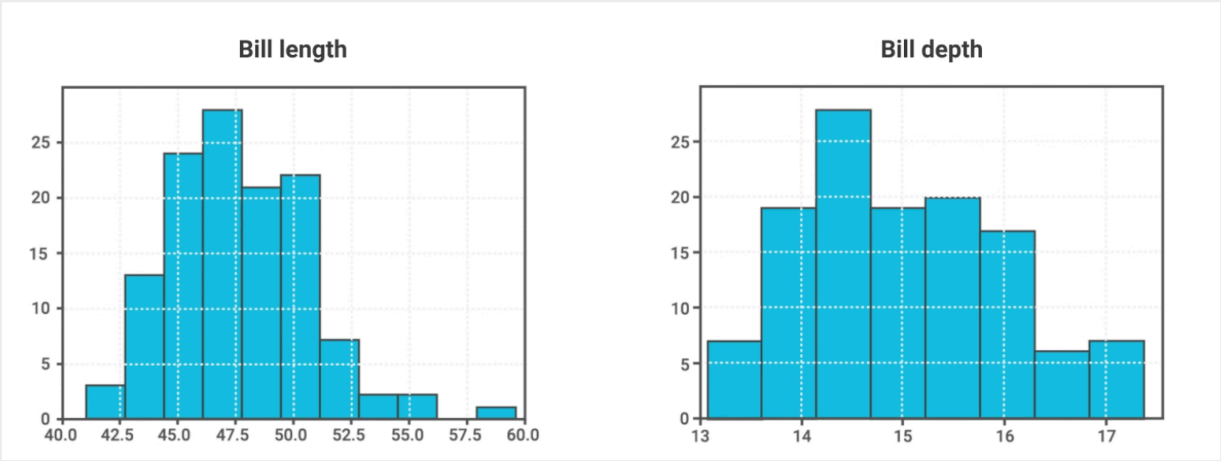


We can load this data into a pandas dataframe and easily create histograms for the two columns, Bill length and Bill depth.

	Bill length (*L*)	Bill depth (*D*)
68	46.1	13.2
69	50.0	16.3
70	48.7	14.1
71	50.0	15.2
72	47.6	14.5
...	...	...
186	47.2	13.7
188	46.8	14.3
189	50.4	15.7
190	45.2	14.8
191	49.9	16.1

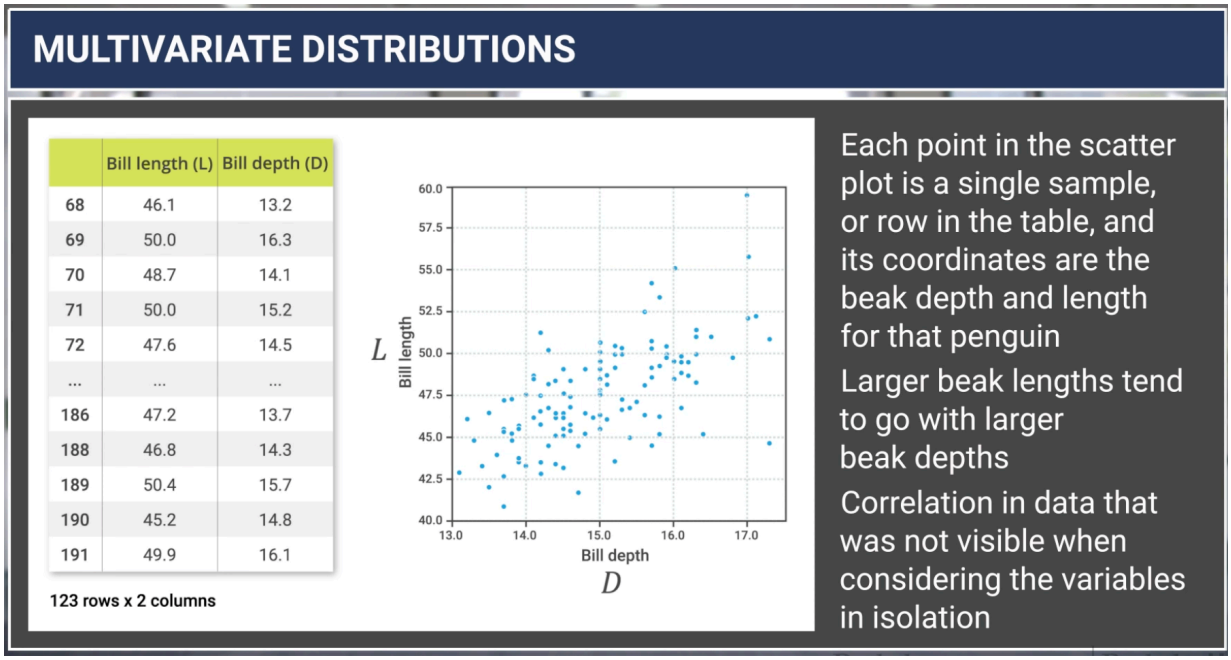
123 rows x 2 columns

These histograms show us that the mean bill length is around 47mm and the mean bill depth is around 15 mm.



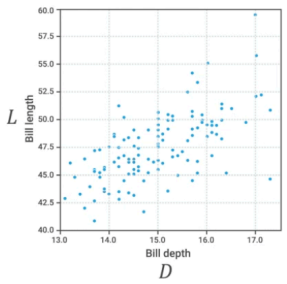
Let's look at a scatterplot of this data. Each point in the scatterplot is a single sample or row in the table, and its coordinates are the beak depth and length for that penguin.

In this view, we can recognize a trend, larger beak lengths tend to go with larger beak depths. There is a correlation in this data that was not visible when we considered the variables in isolation.



To preserve this relationship, we must use a multivariate random variable. A multivariate random variable is simply a collection of random variables, in this case, capital D, the Bill depth, and capital L, the Bill length, arranged into a vector and endowed with a joint probability distribution. When we sample a multivariate random variable, we get a vector instead of a scalar.

MULTIVARIATE DISTRIBUTIONS



Multivariate random variable: $X = \begin{bmatrix} D \\ L \end{bmatrix}$

Joint distribution: $f_X(d, l)$

Sample: $X \rightarrow \begin{bmatrix} d \\ l \end{bmatrix}$

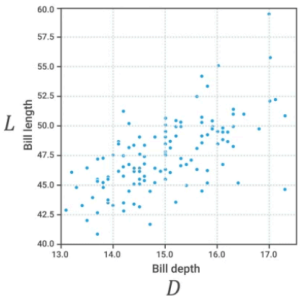
A multivariate random variable is a collection of random variables

Bill depth (D) and bill length (L) are arranged into a vector and endowed with a joint probability distribution

Get a vector (vs scalar) when sampling a multivariate random variable

But all of the concepts that we have studied thus far for single random variables will extend easily to the multivariate case.

The concept of expectation is the same. The mean of a multivariate random variable is the vector of the means of the individual components.



Multivariate random variable: $X = \begin{bmatrix} D \\ L \end{bmatrix}$

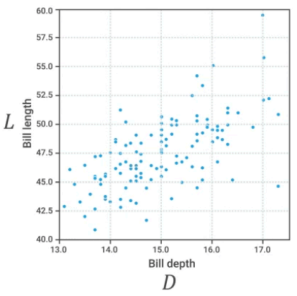
Joint distribution: $f_X(d, l)$

Sample: $X \rightarrow \begin{bmatrix} d \\ l \end{bmatrix}$

Expectation: $E[X] = \begin{bmatrix} E[D] \\ E[L] \end{bmatrix}$

The mean of a multivariate random variable is the vector of the means of the individual components

So, it is still the center of mass of the pdf. The law of large numbers still applies. The mean of a sample of size n will converge to the true mean as the sample size becomes large.



Multivariate random variable: $X = \begin{bmatrix} D \\ L \end{bmatrix}$

Joint distribution: $f_X(d, l)$

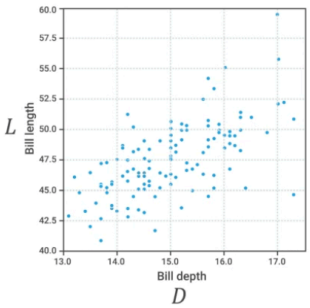
Sample: $X \rightarrow \begin{bmatrix} d \\ l \end{bmatrix}$

Expectation: $E[X] = \begin{bmatrix} E[D] \\ E[L] \end{bmatrix}$

Law of large numbers: $\bar{\mu}_n \rightarrow E[X] \; as \; n \rightarrow \infty$

The mean of a sample of size *n* will converge to the true mean as the sample size becomes large

Keep in mind though that these are all now vector quantities and the central limit theorem remains unchanged.



Multivariate random variable: $X = \begin{bmatrix} D \\ L \end{bmatrix}$

Joint distribution: $f_X(d, l)$

Sample: $X \rightarrow \begin{bmatrix} d \\ l \end{bmatrix}$

Expectation: $E[X] = \begin{bmatrix} E[D] \\ E[L] \end{bmatrix}$

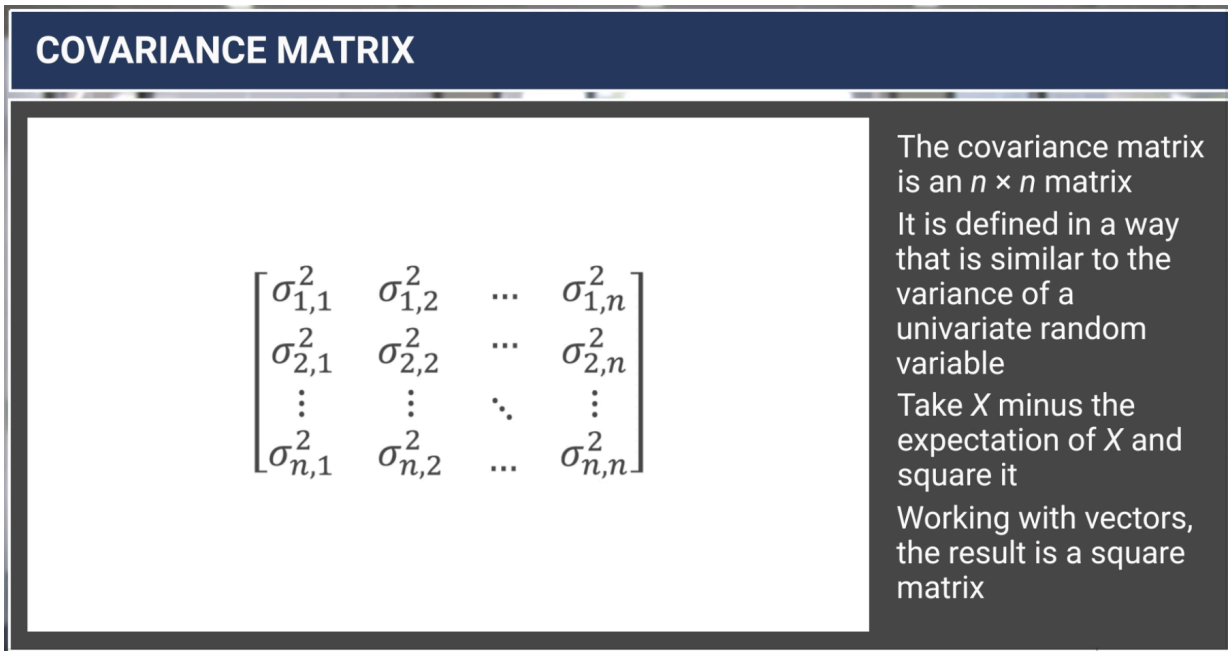
Law of large numbers: $\bar{\mu}_n \rightarrow E[X] \; as \; n \rightarrow \infty$

Central limit theorem: $\bar{X}_n \rightarrow \mathcal{N}(\mu_X, \sigma_X^2/n) \; as \; n \rightarrow \infty$

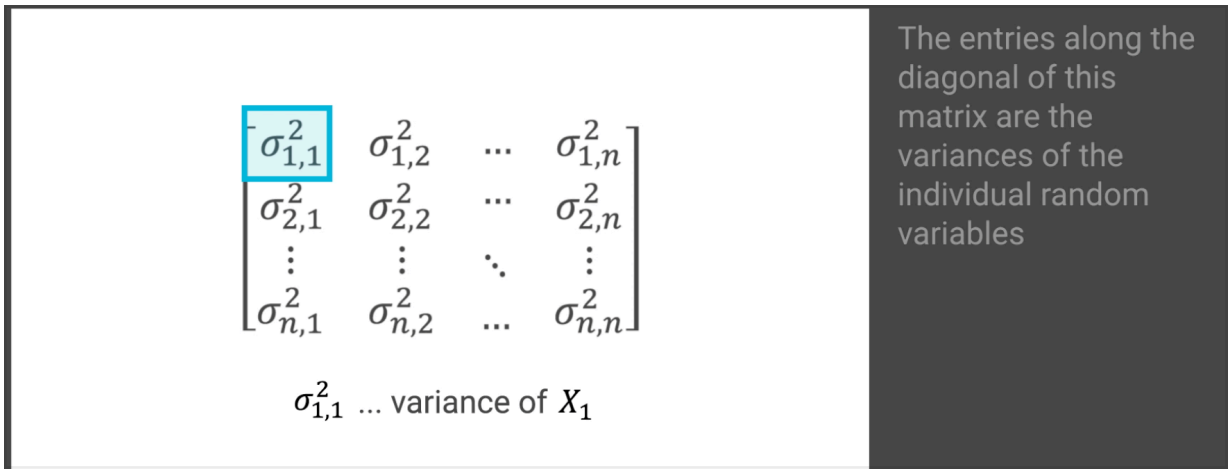
These are all vector quantities; the central limit theorem remains unchanged

The concept of variance, however, changes a bit. Instead of a scalar quantity called variance, we now have a matrix quantity called the covariance matrix.

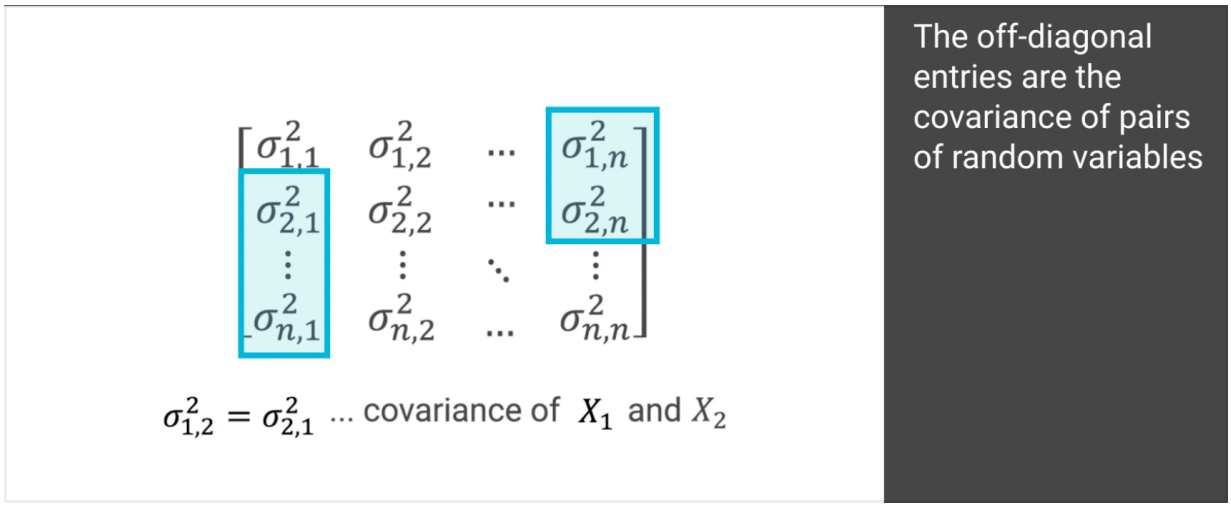
To introduce the covariance matrix, consider a multivariate random variable, not with two entries, but with n entries, X_1 through X_n. The covariance matrix is an n by n matrix, and it is defined in a way that is similar to the variance of a univariate random variable, by taking X minus the expectation of X, and squaring it. Except that now, since we are working with vectors, the result is a square matrix.



The entries along the diagonal of this matrix are the variances of the individual random variables. So, for example, sigma 1,1 squared is the variance of X_1.



Recall that the variance is a measure of the spread of a distribution. The off-diagonal entries are the covariance of pairs of random variables. So, sigma 1,2 squared is the covariance of X_1 and X_2. The covariance of two random variables is a measure of their tendency to vary together. This is the information that we were losing when we considered the variables in isolation.

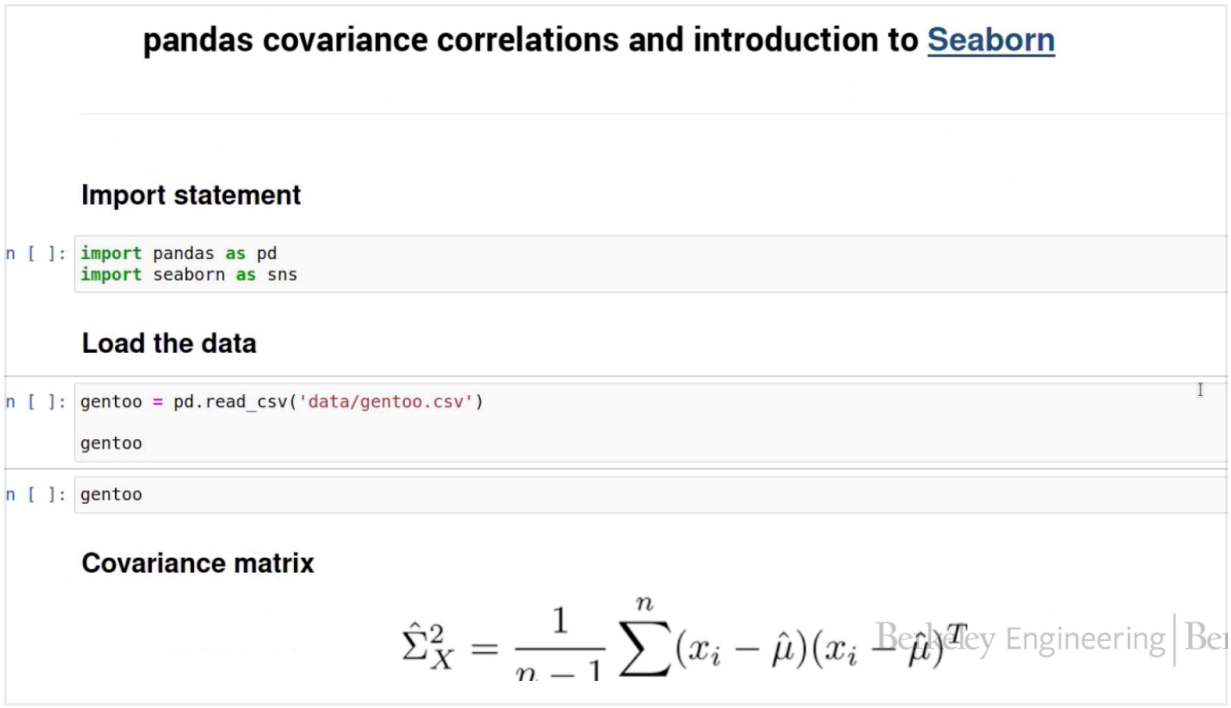


We will see how the covariances are related to the correlation between two quantities. The covariance between X_1 and X_2 is the same as the covariance between X_2 and X_1 . So the covariance matrix has an axis of symmetry along its diagonal. Finally, just as we use a lowercase sigma for the variance, we often use a capital letter sigma as shorthand for the covariance matrix.

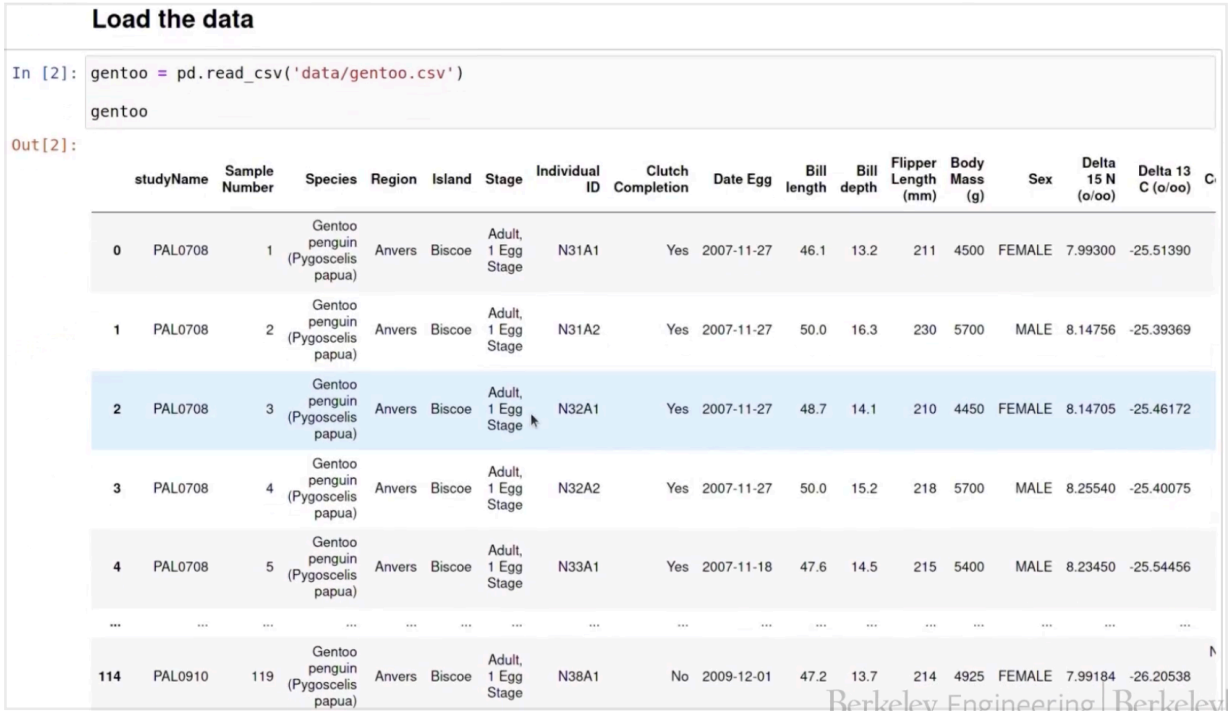
Let's return now to the Gentoo example. The dataset should be understood as a 123-size sample from a multivariate distribution of all Gentoo penguins, which remains hidden. From the data, however, we can estimate the covariance matrix by computing the sample covariance, sigma-hat. Here's the formula for the sample covariance matrix and you can see the similarity to the true covariance matrix. In practice, we will not code this formula explicitly, but instead we will use Python methods. Let's look at the code for computing the sample covariance matrix.

Video 5: Covariance

Now we will use pandas to compute covariance and correlation matrices from data. We call these the sample covariance and correlation matrices to distinguish them from the theoretical ones, which are unknown. And we will also be introducing the Seaborn package for plotting. Seaborn is built on Matplotlib, but it provides a lot of really beautiful plots that are useful for data science, so it's often preferred. And we'll be using that here.



The first thing we'll do is import, and the common alias for Seaborn is sns. So, that's what we'll use. And here we'll load our Gentoo data. We see that the Gentoo data is kind of big.



	studyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Bill length	Bill depth	Flipper Length (mm)	Body Mass (g)	Sex	Delta 15 N (o/oo)	Delta 13 C (o/oo)	C
0	PAL0708	1	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N31A1	Yes	2007-11-27	46.1	13.2	211	4500	FEMALE	7.99300	-25.51390	
1	PAL0708	2	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N31A2	Yes	2007-11-27	50.0	16.3	230	5700	MALE	8.14756	-25.39369	
2	PAL0708	3	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N32A1	Yes	2007-11-27	48.7	14.1	210	4450	FEMALE	8.14705	-25.46172	
3	PAL0708	4	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N32A2	Yes	2007-11-27	50.0	15.2	218	5700	MALE	8.25540	-25.40075	
4	PAL0708	5	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N33A1	Yes	2007-11-18	47.6	14.5	215	5400	MALE	8.23450	-25.54456	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
114	PAL0910	119	Gentoo penguin (Pygoscelis papua)	Anvers	Biscoe	Adult, 1 Egg Stage	N38A1	No	2009-12-01	47.2	13.7	214	4925	FEMALE	7.99184	-26.20538	

It has 17 columns and we don't need all of this information for what we're doing here. So we're just going to keep these columns, these four columns, Bill length,

Bill depth, Flipper Length, and Body Mass. So let's take those columns, and we're going to make strings of the names, and then put commas in there, and assign that to override the Gentoo dataframe. And so now we have a smaller dataframe.

Load the data

```
In [2]: gentoo = pd.read_csv('data/gentoo.csv')
        gentoo = gentoo[['Bill length','Bill depth','Flipper Length (mm)','Body Mass (g)']]
```

Out[2]:

ıdyName	Sample Number	Species	Region	Island	Stage	Individual ID	Clutch Completion	Date Egg	Bill length	Bill depth	Flipper Length (mm)	Body Mass (g)	Sex	Delta 15 N (o/oo)	Delta 13 C (o/oo)	Comments

OK, the first thing we want to do is compute a covariance matrix. This is the formula for the covariance matrix that we saw in the lecture, and in pandas, it's really straightforward. All we have to do is call cov on it and it will compute the covariance matrix.

Covariance matrix

$$\hat{\Sigma}_X^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T$$

```
In [4]: gentoo.cov()
```

Out[4]:

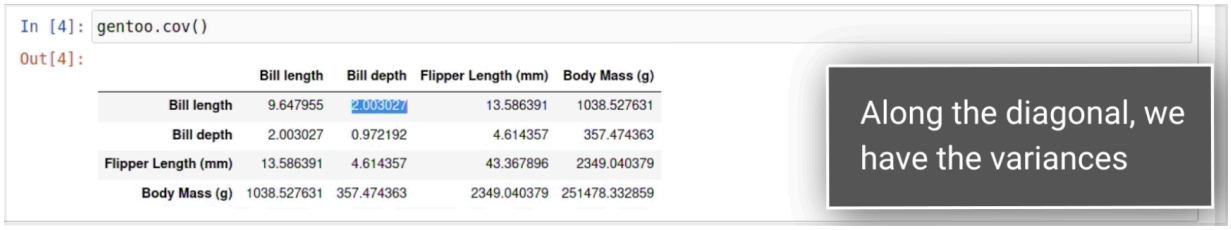
	Bill length	Bill depth	Flipper Length (mm)	Body Mass (g)
Bill length	9.647955	2.003027	13.586391	1038.527631
Bill depth	2.003027	0.972192	4.614357	357.474363
Flipper Length (mm)	13.586391	4.614357	43.367896	2349.040379
Body Mass (g)	1038.527631	357.474363	2349.040379	251478.332859

Recall that the covariance matrix is symmetric, meaning that the covariance between, say Bill depth and Bill length is 2.00, is the same as the covariance between Bill length and Bill depth on the other side of the matrix. So, it has an axis of symmetry about the diagonal.

	Bill length	Bill depth	Flipper Length (mm)	Body Mass (g)
Bill length	9.647955	2.003027	13.586391	1038.527631
Bill depth	2.003027	0.972192	4.614357	357.474363
Flipper Length (mm)	13.586391	4.614357	43.367896	2349.040379
Body Mass (g)	1038.527631	357.474363	2349.040379	251478.332859

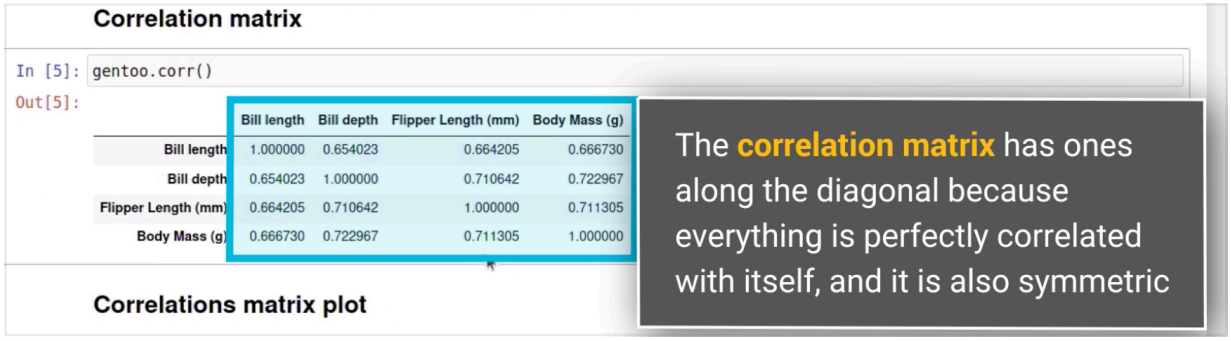
And this is telling us that there is covariance between all of these, but the scales

are quite different. Along the diagonal, we have the variances and even though the covariance between the Flipper Length and the Body Mass seems huge, we also see that the variance of Body Mass is really large.



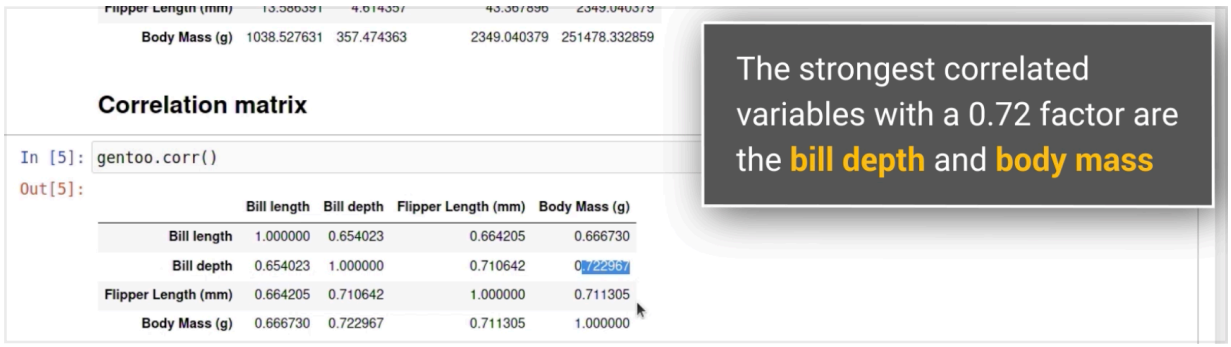
	Bill length	Bill depth	Flipper Length (mm)	Body Mass (g)
Bill length	9.647955	2.003027	13.586391	1038.527631
Bill depth	2.003027	0.972192	4.614357	357.474363
Flipper Length (mm)	13.586391	4.614357	43.367896	2349.040379
Body Mass (g)	1038.527631	357.474363	2349.040379	251478.332859

So, that's a little bit difficult to interpret. It's easier to interpret a correlation matrix. So that's also very easy to produce in pandas, and there it is.



	Bill length	Bill depth	Flipper Length (mm)	Body Mass (g)
Bill length	1.000000	0.654023	0.664205	0.666730
Bill depth	0.654023	1.000000	0.710642	0.722967
Flipper Length (mm)	0.664205	0.710642	1.000000	0.711305
Body Mass (g)	0.666730	0.722967	0.711305	1.000000

The correlation matrix, as we know, has ones along the diagonal because everything is perfectly correlated with itself, and then it's also symmetric. So we see a 0.65 here, and a 0.65 here, and we see that there are somewhat strong correlations between all of these random variables. The strongest correlated are the bill depth and the body mass with a correlation factor of 0.72.



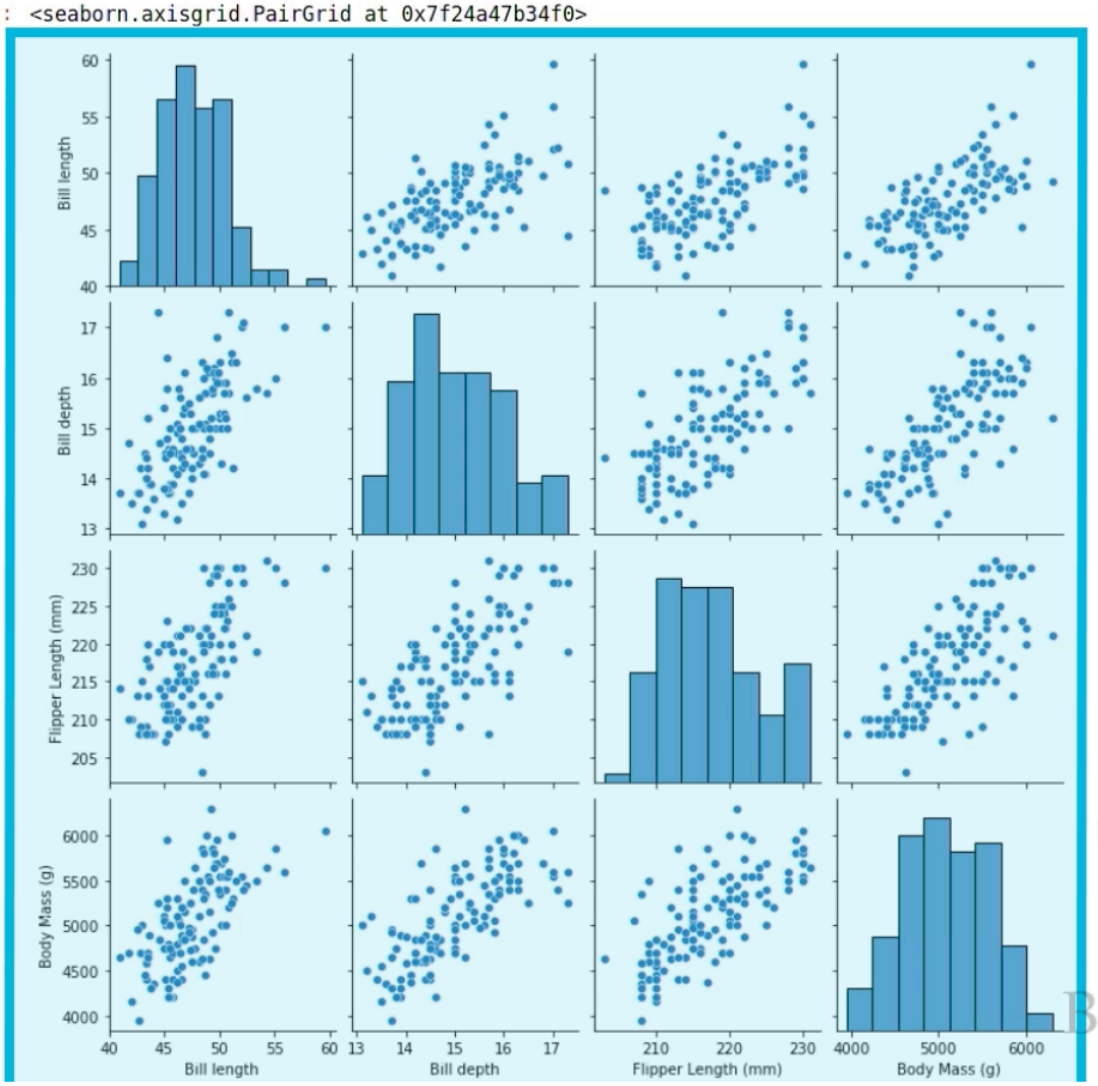
	Bill length	Bill depth	Flipper Length (mm)	Body Mass (g)
Bill length	1.000000	0.654023	0.664205	0.666730
Bill depth	0.654023	1.000000	0.710642	0.722967
Flipper Length (mm)	0.664205	0.710642	1.000000	0.711305
Body Mass (g)	0.666730	0.722967	0.711305	1.000000

So, that's helpful to know, and it may come in handy when we start building machine learning models for, say, predicting Body Mass from Flipper Length, or predicting something else from these four quantities. A correlation matrix is often a good place to start.

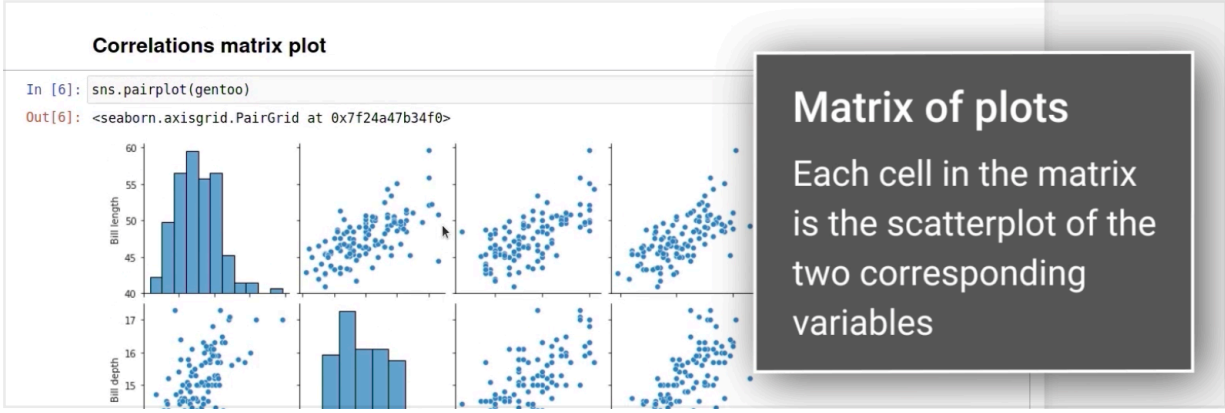
Another thing that I want to show you is the correlations matrix plot that is provided by Seaborn, and that is called in Seaborn a pair plot. So all we have to do is call pairplot on the Gentoo dataframe, and Seaborn will produce this nice picture, which is essentially the same information as the correlations matrix.

Correlations matrix plot

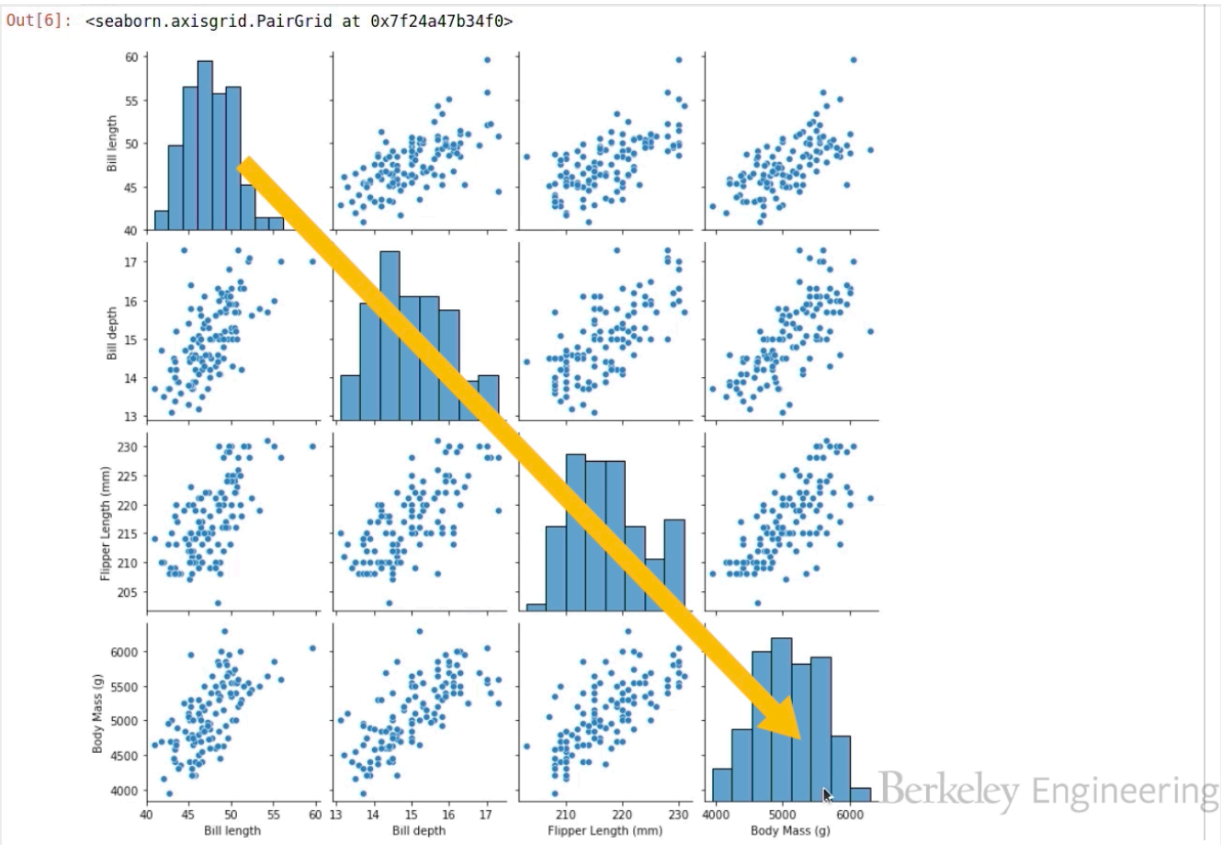
```
In [6]: sns.pairplot(gentoo)
Out[6]: <seaborn.axisgrid.PairGrid at 0x7f24a47b34f0>
```



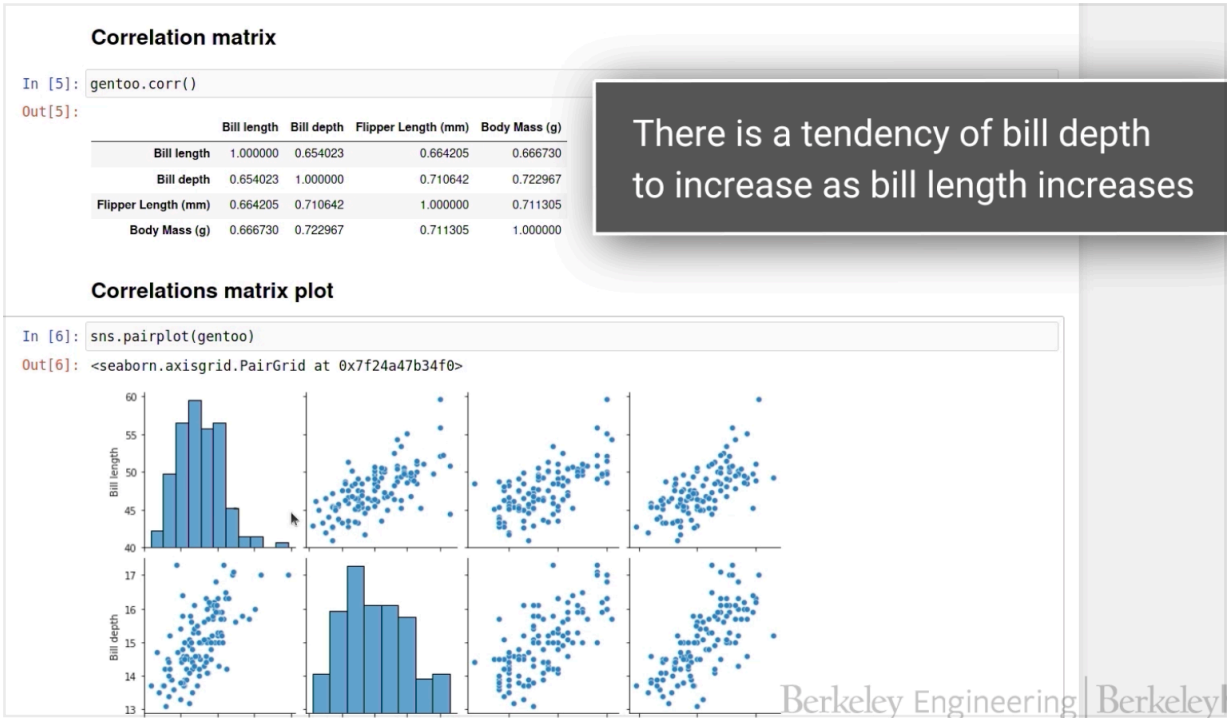
It's a four by four matrix of plots and each cell in the matrix is the scatterplot of the two corresponding variables. So, this scatterplot in the 1,2 cell is a scatterplot between Bill length and Bill depth.



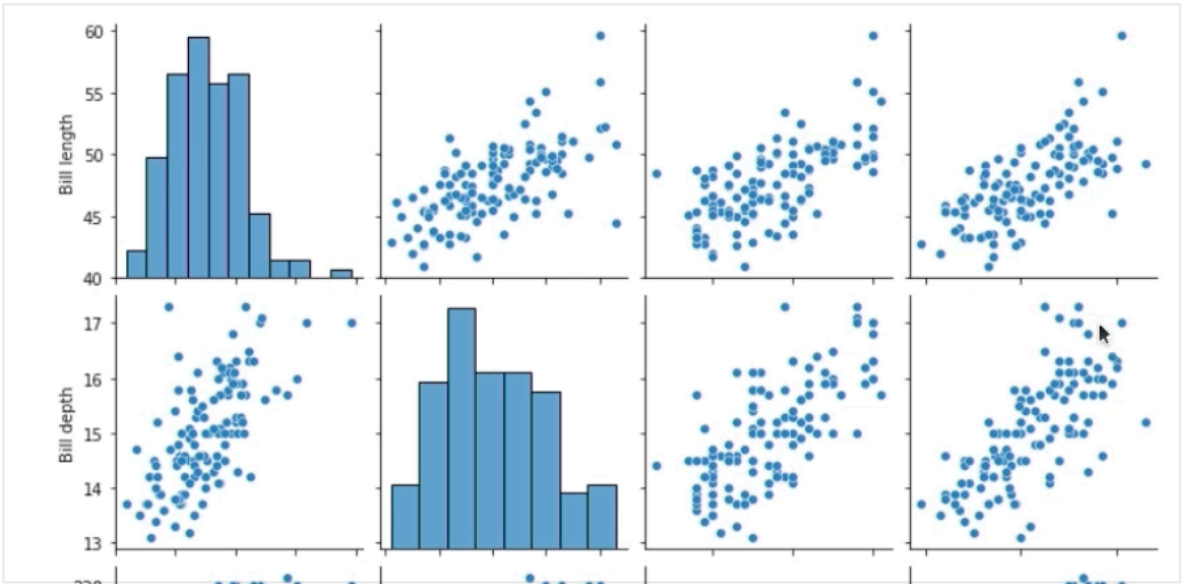
So, here we can see the correlations a little bit more directly. Along the diagonal, it doesn't include scatterplots because it would be useless to see a scatterplot of Bill length versus Bill length.



Instead, it includes a histogram. And what we can observe here are these correlations in action. This scatterplot is a correlation of 0.65. So we see a tendency of the bill depth to increase as the bill length increases.



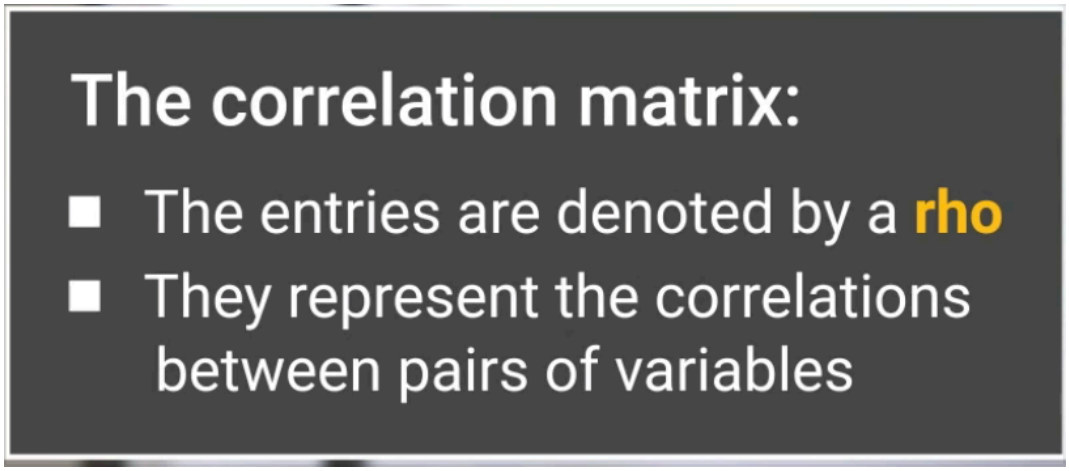
But the strongest correlation here would be bill depth versus bill mass, and that is this scatterplot in the 2,4 location, which is the same as the scatterplot in the 4,2 location, except flipped along a diagonal.



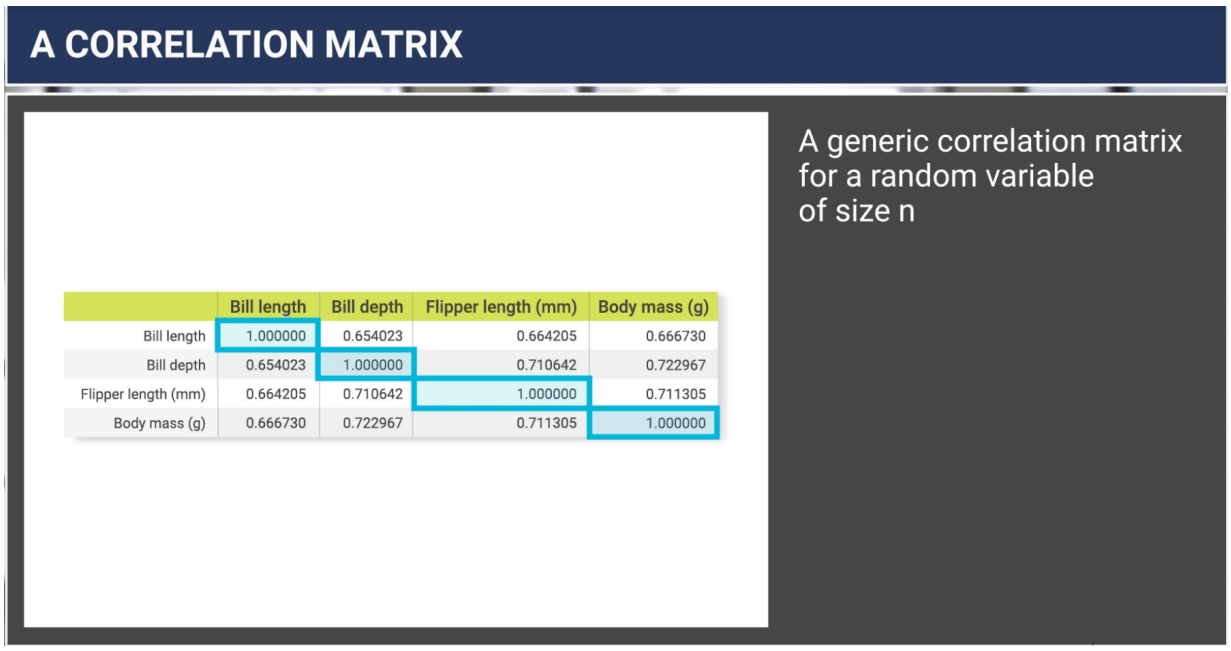
So, this is also a symmetric grid of plots.

Video 6: Correlation, Conditional Probabilities, and Independence

The entries in the correlation matrix are denoted by a rho, and they represent the correlations between pairs of variables.



Here's the correlation matrix for our Gentoo data. And here we see a generic correlation matrix for a random variable of size n. The diagonal entries are all 1, because a variable is always perfectly correlated with itself.



	Bill length	Bill depth	Flipper length (mm)	Body mass (g)
Bill length	1.000000	0.654023	0.664205	0.666730
Bill depth	0.654023	1.000000	0.710642	0.722967
Flipper length (mm)	0.664205	0.710642	1.000000	0.711305
Body mass (g)	0.666730	0.722967	0.711305	1.000000

The i_j entry in the correlation matrix is obtained by taking the i_j covariance and dividing it by the standard deviations of X_i and X_j.

	Bill length	Bill depth	Flipper length (mm)	Body mass (g)
Bill length	1.000000	0.654023	0.664205	0.666730
Bill depth	0.654023	1.000000	0.710642	0.722967
Flipper length (mm)	0.664205	0.710642	1.000000	0.711305
Body mass (g)	0.666730	0.722967	0.711305	1.000000

$$Corr[X] = \begin{bmatrix} 1 & \rho_{1,2} & \cdots & \rho_{1,n} \\ \rho_{2,1} & 1 & \cdots & \rho_{2,n} \\ \vdots & & & \\ \rho_n & & & \end{bmatrix}$$

$$\rho_{i,j} = \frac{\sigma_{i,j}^2}{\sigma_i \sigma_j} \in [-1, 1]$$

A generic correlation matrix for a random variable of size n

The i_j entry is obtained by taking the i_j covariance and dividing it by the standard deviations of X_i and X_j

It can be shown that the result is between –1 and 1. Like the covariance, the correlation tells us something about the tendency of two variables to vary together.

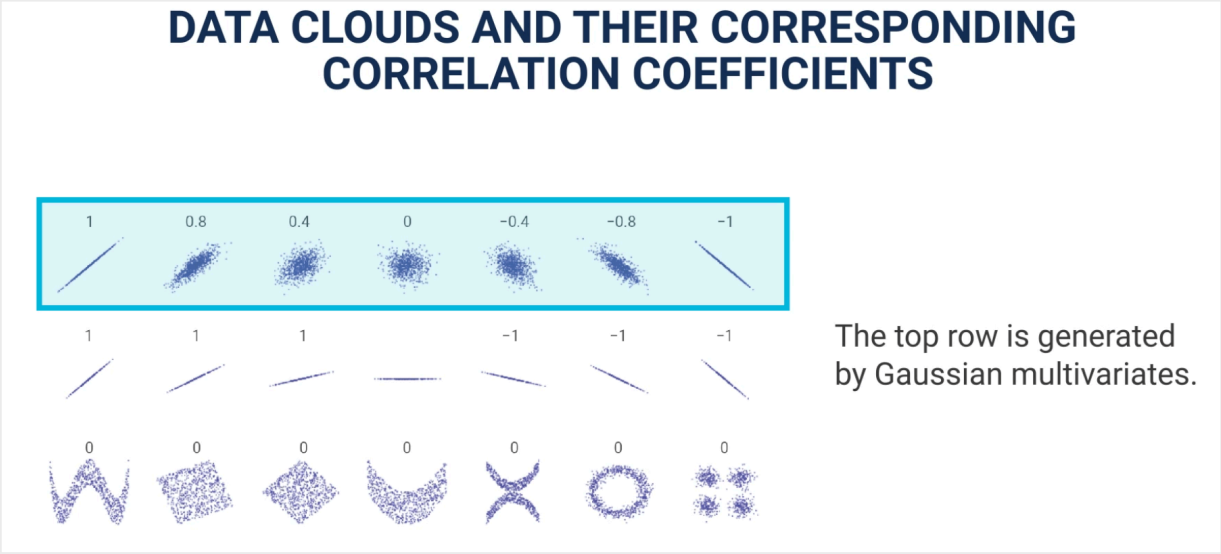
A negative correlation means that when one goes up, the other tends to go down. And when one goes down, the other tends to go up. A positive correlation means that they both go up and down together. And a zero correlation means that neither of these is true.

The correlation tells you about the tendency of two variables to vary together

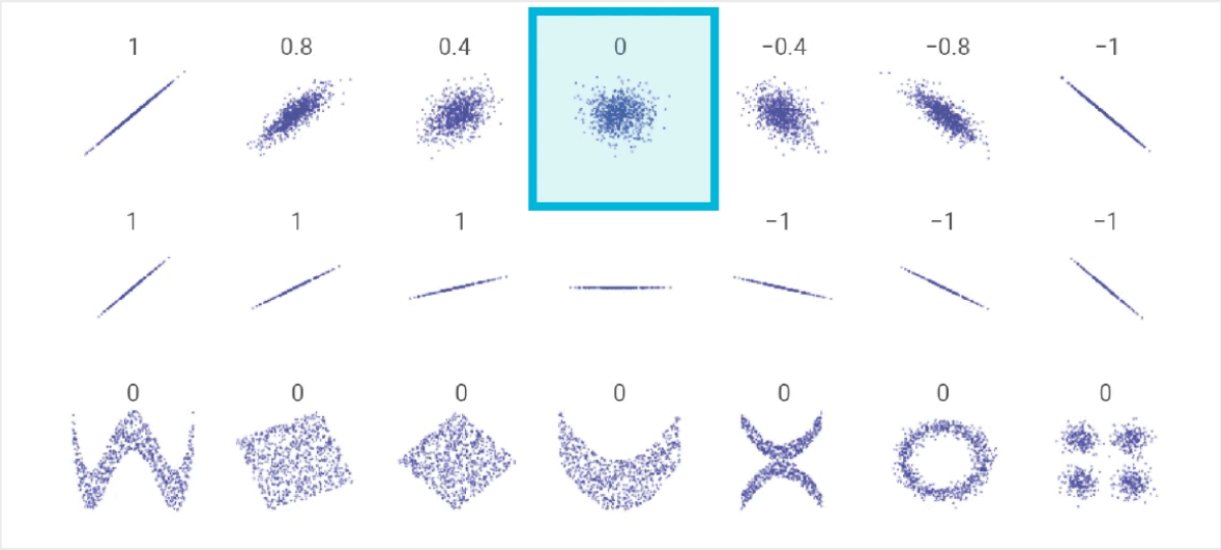
A negative correlation means that when one goes up, the other tends to go down

A positive correlation means that they both go up and down together

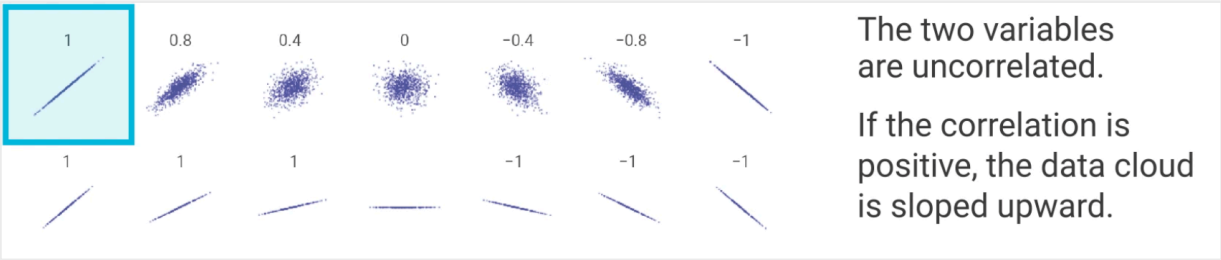
Here's a handy diagram of data clouds and their corresponding correlation coefficients. The top row is generated by Gaussian multivariates.



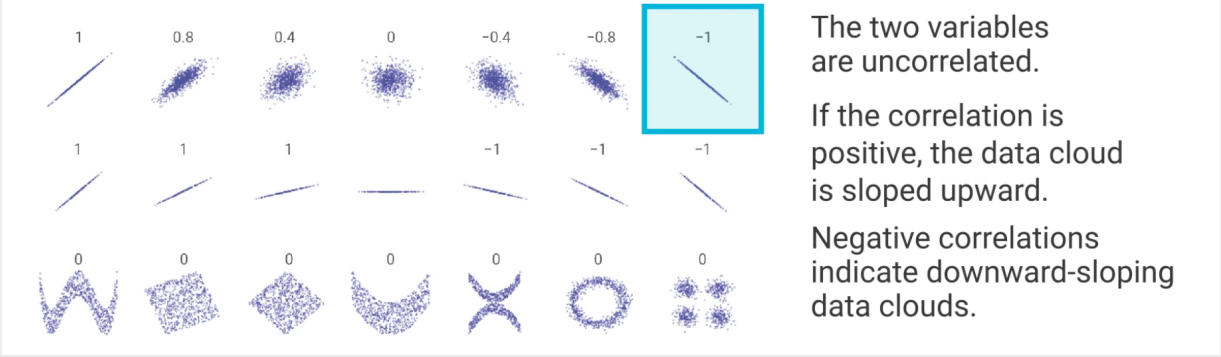
In the middle plot of the top row, the two variables are uncorrelated.



When the correlation is positive, the data cloud is sloped upward. And the closer to 1, the more the data coalesces into a line.



Negative correlations indicate downward-sloping data clouds.

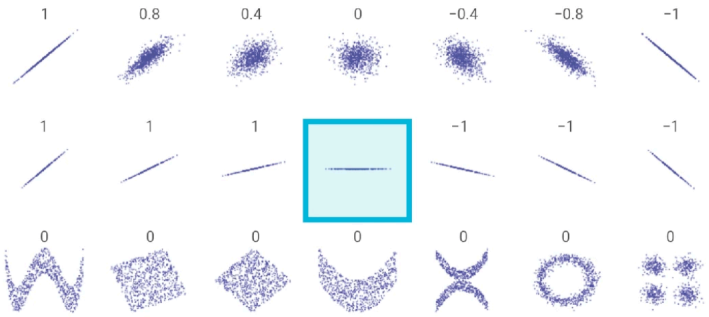


The middle row shows cases of perfect correlation, where rho_ij equals 1 or –1.

The middle row shows perfect correlation where rho_{ij} = 1 or -1.

In the middle plot here, the correlation coefficient is undefined, since one of the standard deviations is zero.

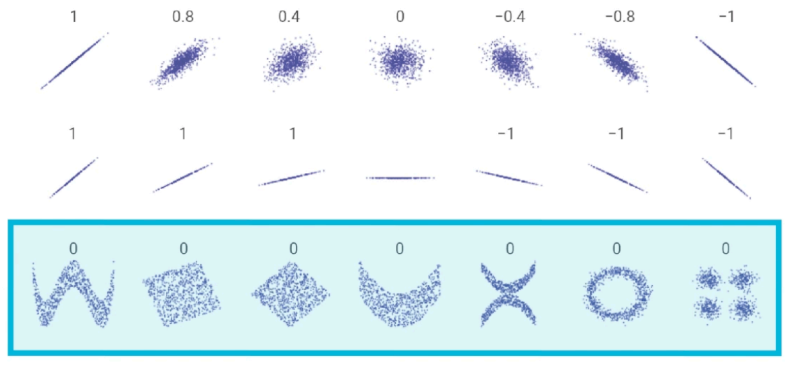
These plots are making the point that the correlation is not capturing the slope of the data.



The correlation coefficient is undefined since one of the standard deviations is zero.

The correlation is not capturing the slope of the data.

All upward-sloping plots have correlation equal to 1, and all downward-sloping plots have correlation equal to –1. The bottom row shows various examples of uncorrelated data. And we see that the lack of correlation does not mean that the variables are not predictive of each other.

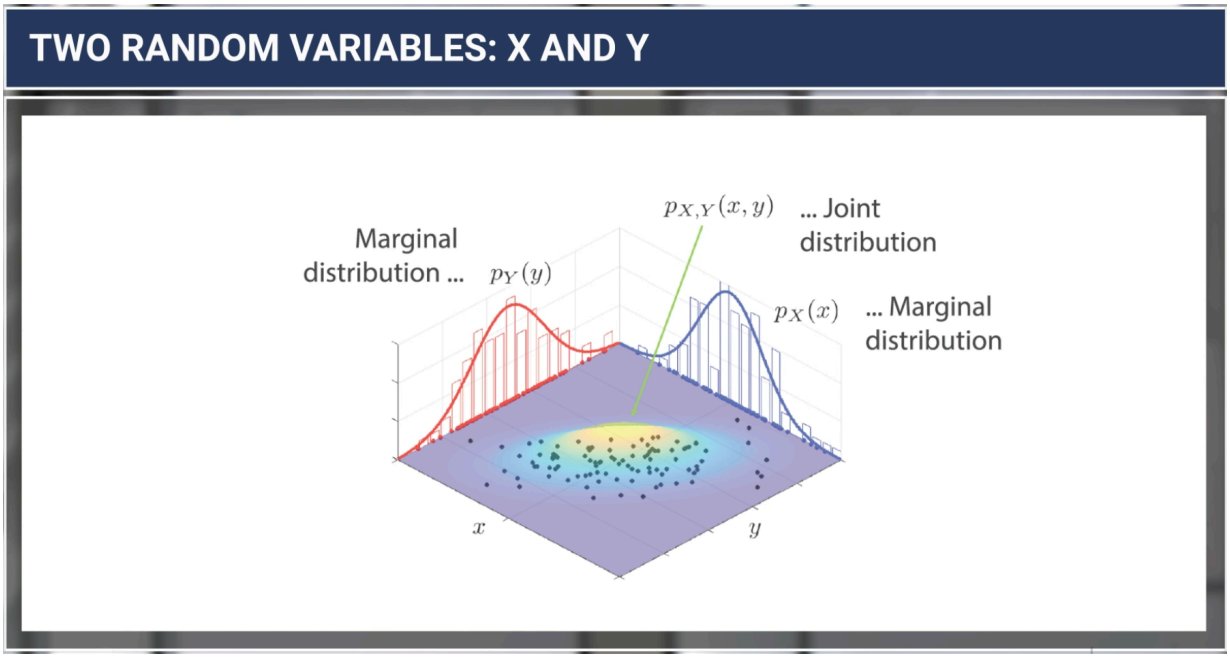


The bottom row shows various examples of uncorrelated data.

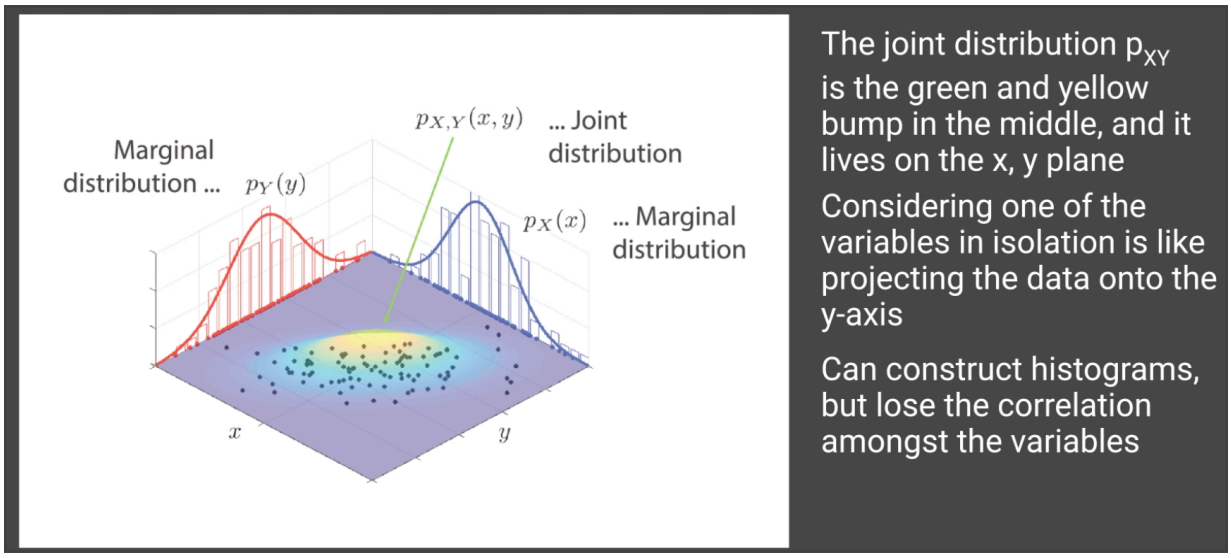
Here on the bottom left, for example, we see the data seems to follow a sine wave. So, knowing the value of x gives us a good sense of the value of y.

And yet, these two have zero correlation. In all of these cases, there is no tendency of one variable to go up or down whenever the other goes up or down. These variables, although predictive of each other, do not obey a simple linear or monotonic relationship; hence, they are uncorrelated, but not independent. We will get to the concept of independence of random variables. But first, we need to understand conditional probabilities.

Let's pause for a recap. This picture shows two random variables, X and Y.



These could be, for example, the length and depth of Gentoo beaks. The joint distribution, p_XY, is the green and yellow bump in the middle, and it lives on the x-y plane. It is from this distribution that we sample when we collect data, such as this cloud of black dots. Considering one of the variables in isolation, say "y", is like projecting the data onto the y-axis. When we do that, we get the red dots.

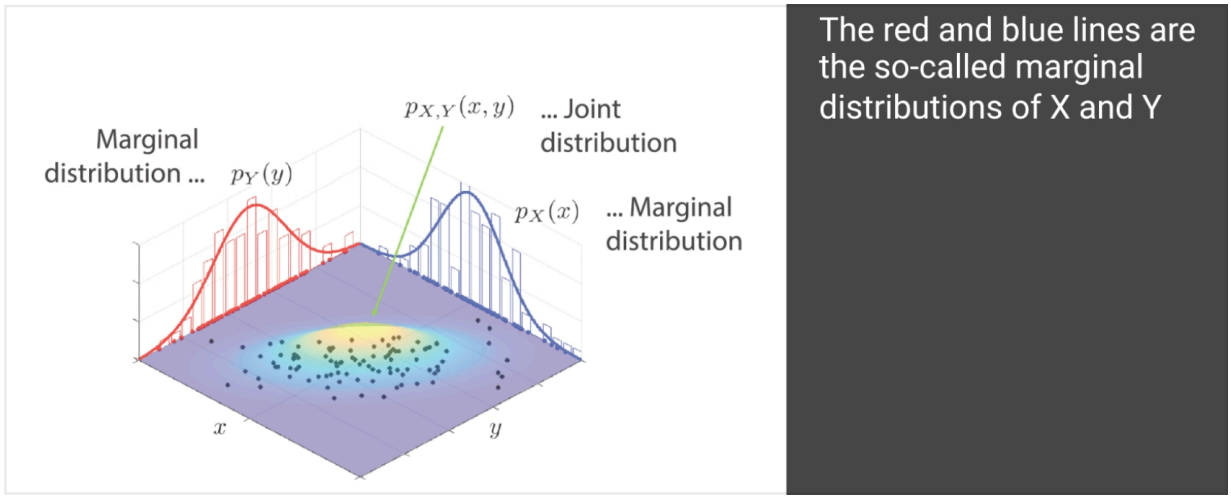


The joint distribution p_{XY} is the green and yellow bump in the middle, and it lives on the x, y plane

Considering one of the variables in isolation is like projecting the data onto the y-axis

Can construct histograms, but lose the correlation amongst the variables

We can then construct histograms with these, as we did with the depth and length of beaks at the beginning of this lecture. However, we then lose the correlation amongst the variables. The red and blue lines are the so-called marginal distributions of X and Y. These are the distributions for the individual variables, X or Y, when we ignore the other. The marginal distributions would produce the blue dots if we were measuring only X, and the red dots if we were measuring only Y.

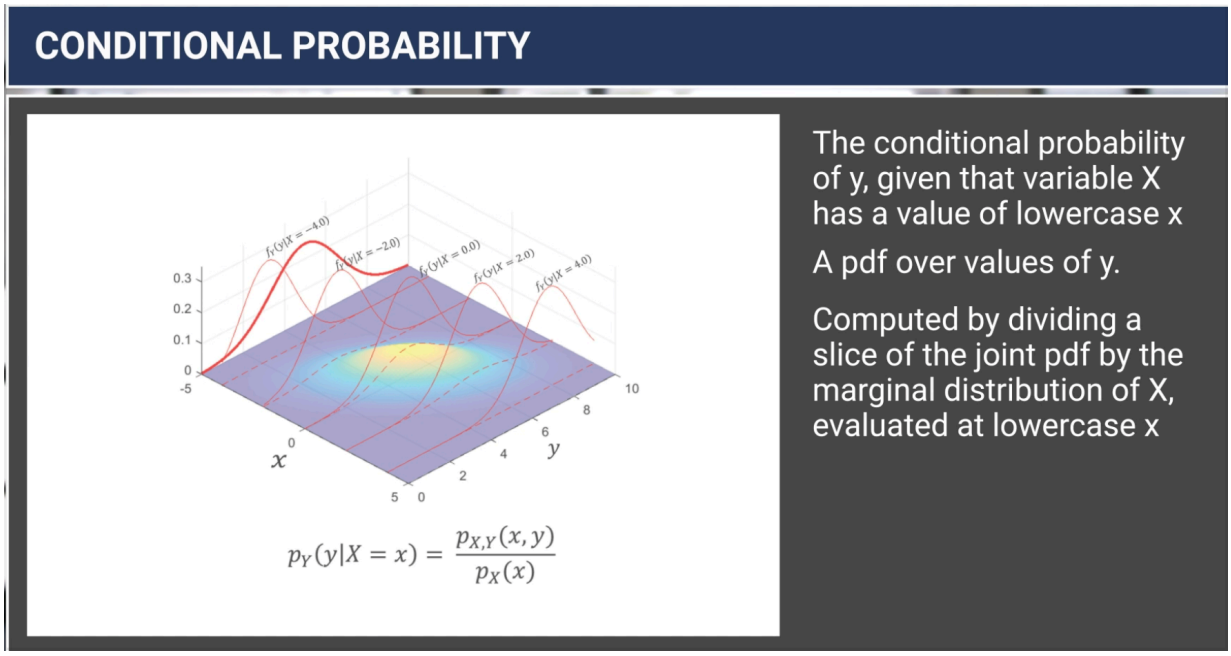


The red and blue lines are the so-called marginal distributions of X and Y

The conditional probability allows us to find distributions in one random variable when the others are fixed at particular values. For example, what is the distribution of Gentoo beak depths amongst birds with beak lengths between 43 and 45 millimeters? This amounts to taking a slice of the joint pdf along the specified value, and then scaling it up so that its integral is one.

The notation for the conditional probability involves a vertical bar. So, we read this as the conditional probability of y, given that the variable X has a value of

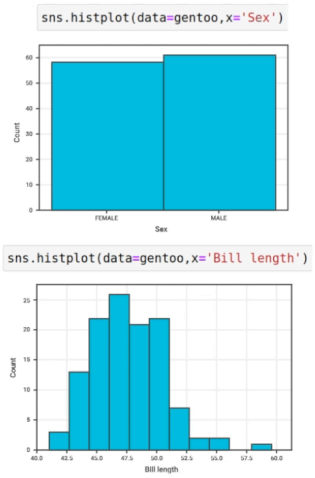
lowercase x. This is a pdf over values of y. And we compute it by dividing the slice of the joint pdf by the marginal distribution of X, evaluated at lowercase x.



This all may seem a little bit abstract, so let's look at an example. Instead of beak lengths and depths, let's instead consider beak lengths and the sex of the bird, male or female. The multivariate variable now contains a continuous quantity, beak lengths, and a discrete quantity, sex. That's fine. A multivariate can contain any collection of random variables continuous and/or discrete.

BEAK LENGTH AND SEX

	Sex	Bill length
0	Female	46.1
1	Male	50.0
2	Female	48.7
3	Male	50.0
4	Male	47.6
...	...	...
114	Female	47.2
115	Female	46.8
116	Male	50.4
117	Female	45.2
118	Male	49.9



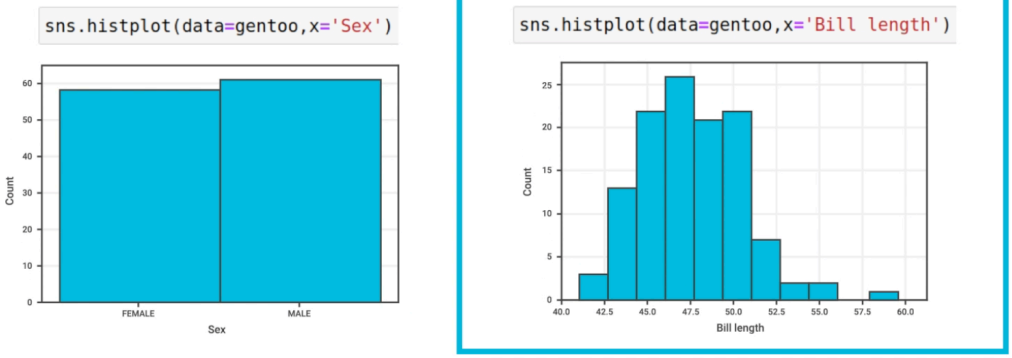
The multivariate variable now contains a continuous quantity, beak/bill lengths, and discrete quantity, sex

A multivariate can contain any collection of random variables, continuous and/or discrete

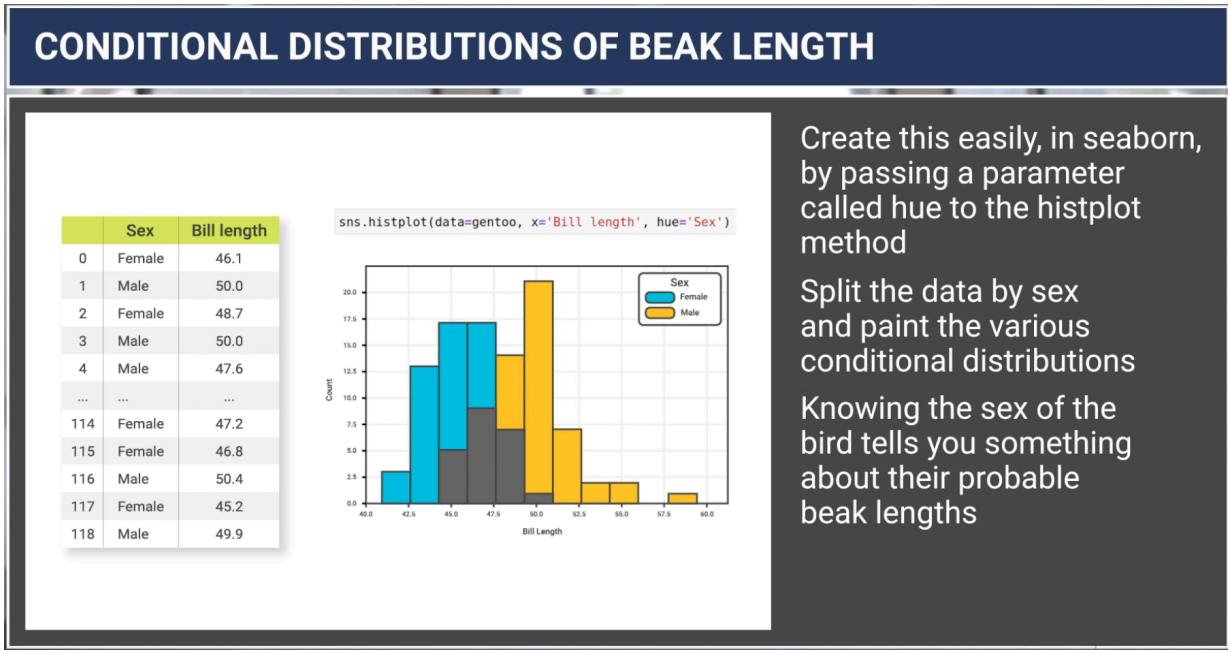
When we look at the histograms, we see that we have about equal proportions of males and females in the dataset. For beak lengths, we get this now familiar distribution.

HISTOGRAM OF BEAK LENGTH AND SEX

You get this now familiar distribution for beak length.



And here are the conditional distributions of beak lengths for male in orange, and female in blue. In Seaborn, you can create this easily by passing a parameter called hue to the histplot method. This tells the program to split the data by sex and paint the various conditional distributions. One thing we notice here is that male Gentoos tend to have longer beaks than females.



In other words, knowing the sex of the bird tells us something about their probable beak lengths. We cannot say that these are correlated, because that concept only applies to numerical values, and we have not yet attached any numbers to the labels, male and female. But what we can say is that beak lengths and sex are dependent random variables.

Let's look more deeply at the concept of dependence of random variables. Two random variables are said to be independent when knowing the value of one tells us nothing about the value of the other. In terms of probability distributions, this means that the distribution of y, given X equals some value, x_1, is the same as the distribution conditioned on X being any other value, x_2.

INDEPENDENCE

$$p_Y(y|X = x_1) = p_Y(y|X = x_2) = p_Y(y)$$

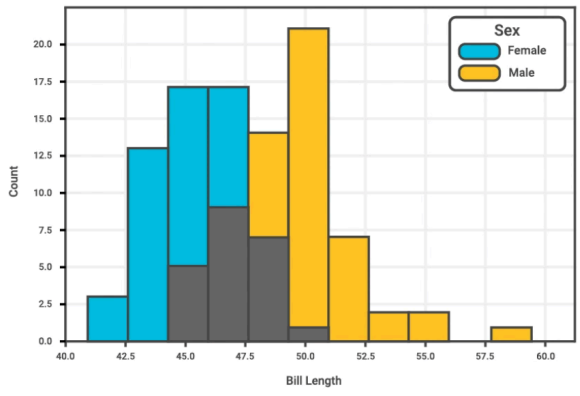
Two random variables are said to be independent when knowing the value of one tells you nothing about the value of the other

Probability distribution of y, given X equals one value x_1, is the same as the distribution conditioned on X being any other value x_2

And also, equal to the marginal distribution of Y. For example, if the distribution of beak lengths for males were identical to the distribution of beak lengths for females, and therefore also equal to the general distribution of beak lengths over the entire population, the marginal distribution, then we would say the sex and beak lengths are independent quantities.

INDEPENDENCE

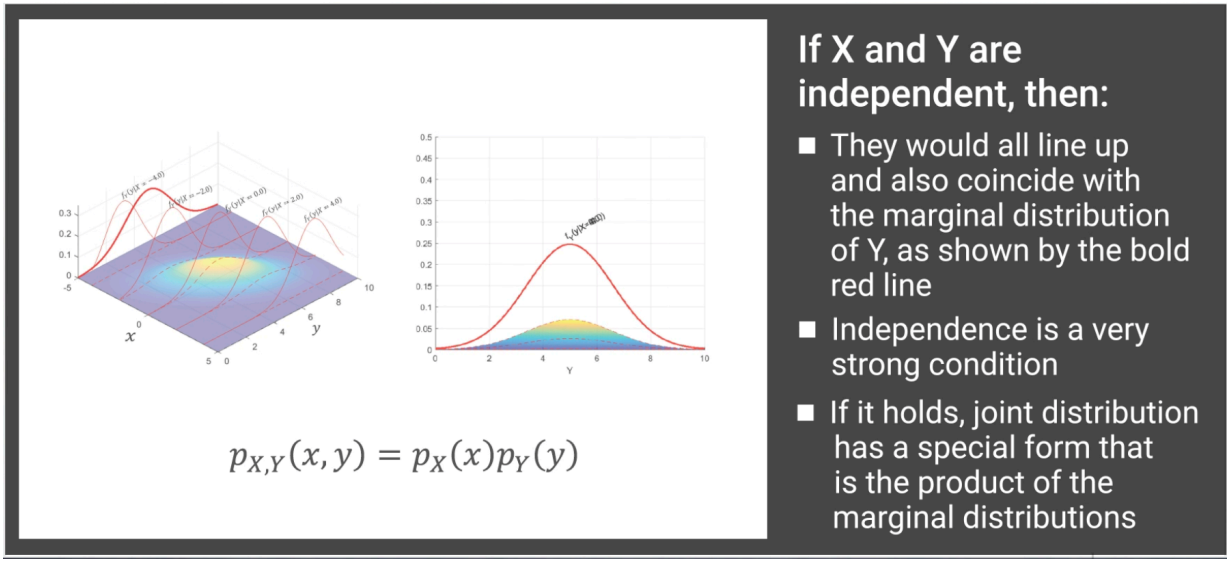
$$p_L(\ell|Sex = male) = p_L(\ell|Sex = female) = p_L(\ell)$$



If the distribution of beak lengths for males and females were identical and equal to the marginal distribution, then you would say that sex and beak length are independent quantities

We can see clearly that this is not the case. The blue and orange distributions are not the same. And so, beak length is not independent of sex.

With two continuous variables, independence means that all conditional distributions are identical. So all slices of the joint distribution parallel to the y-axis and scaled up to a conditional distribution are exactly the same. If we were to look at them head on, as in this picture, they would all line up and also coincide with the marginal distribution of Y, which is shown as the bold red line in this picture.



Independence is a very strong condition. If it holds, then we can also know that the joint distribution has a special form that is the product of the marginal distributions. This is a huge simplification that we will use in future lectures.