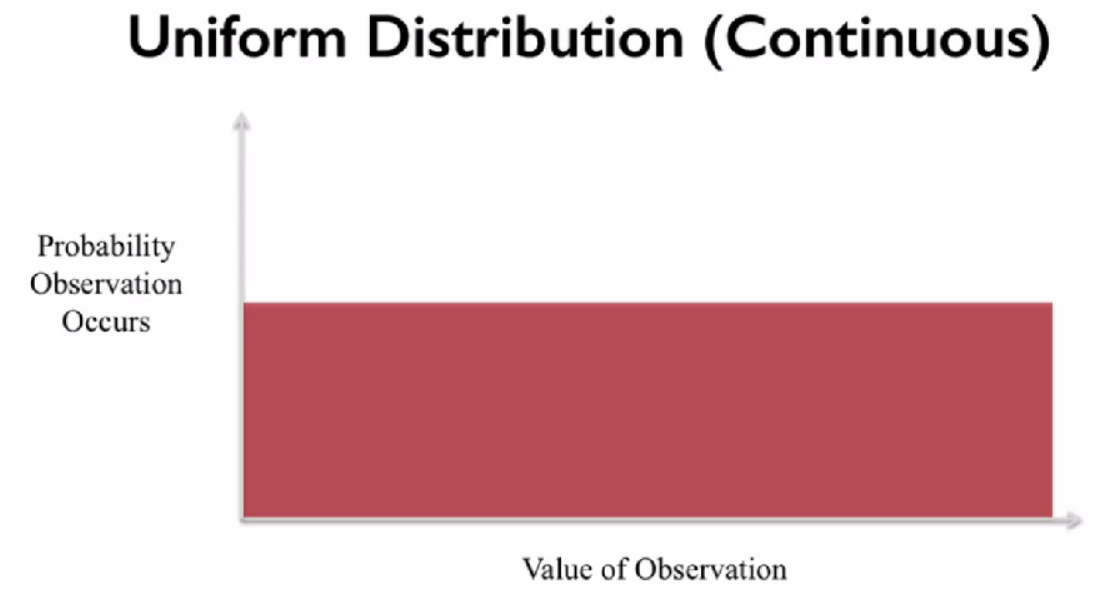
More Distributions

Many of the distributions you use in data science are not discrete binomial, and instead are continues where the value of the given observation isn't a category like heads or tails, but can be represented by a real number.
It's common to then graph these distributions when talking about them, where the x axis is the value of the observation and the y axis represents the probability that a given observation will occur.



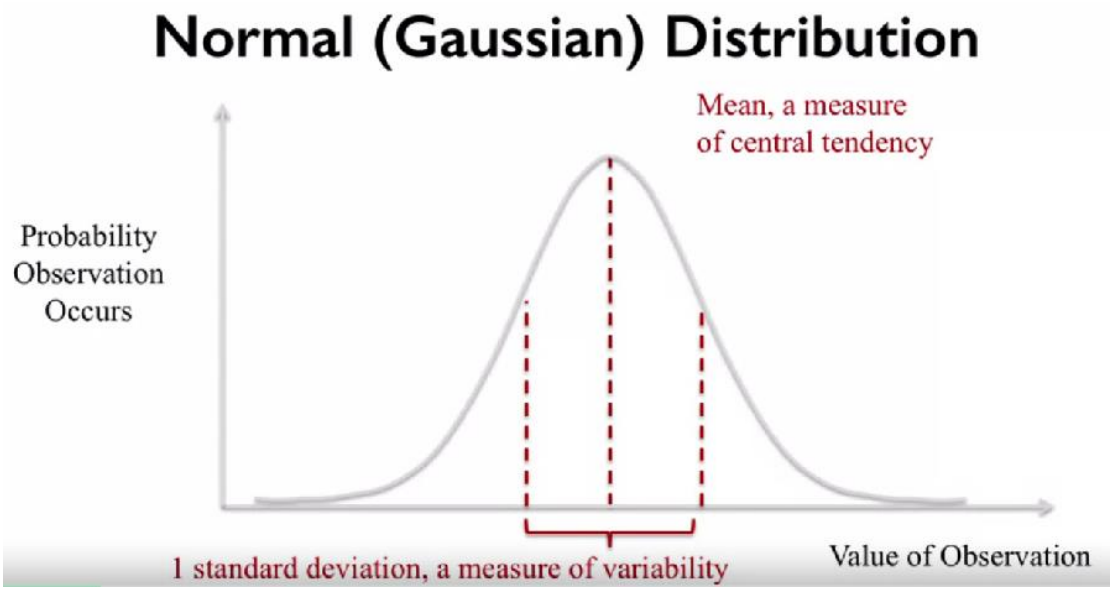
If all numbers are equally likely to be drawn when you sample from, this should be graphed as a flat horizontal line.
And this flat line is actually called the uniform distribution.

```
np.random.uniform(0, 1)
```

```
0.962881285788768
```

The Uniform Distribution (also called the Rectangular Distribution) is the simplest distribution.

It has equal probability for all values of the Random variable.



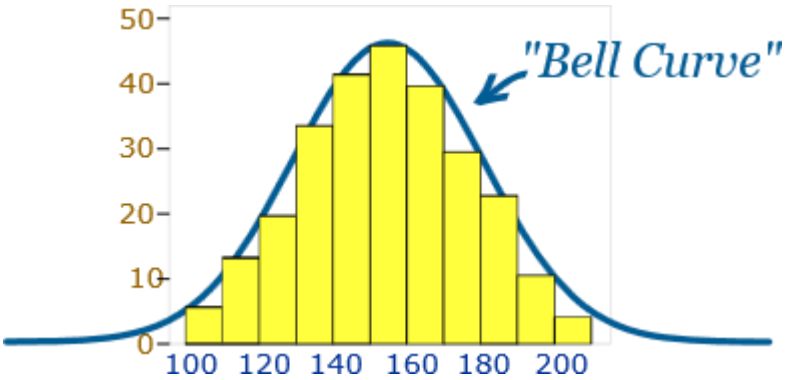
There are few other distributions that get a lot more interesting.
Let's take the normal distribution which is also called Gaussian distribution or sometimes, a Bell Curve.

```
np.random.normal(0.75)
```

```
1.3451976729403343
```

Data can be "distributed" (spread out) in different ways.

- It can be spread out more on the left
- Or more on the right
- Or it can be all jumbled up
- But there are many cases where the data tends to be around a central value with no bias left or right, and it gets close to a "Normal Distribution" like this:



Many things closely follow a Normal Distribution like: heights of people, blood pressure, marks on a test
The Normal Distribution has:

- mean = median = mode
- symmetry about the center
- 50% of values less than the mean
- and 50% greater than the mean

I want to introduce you to the term expected value
The mean:
Is the sum of all the values divided by the total number of values.
Calculating a mean values are computational process, and it takes place by looking at samples from distribution.
For instance rolling a die three times might give you 1, 2 and 6, the mean value is then 3.5.

The expected value is the probability from the underlying distribution is what would be the mean of a die roll if we did an infinite number of rolls.
The result is 3.5 since each face of the die is equally likely to appear.
Thus the expected value is 3.5, while the mean value depends upon the samples that we've taken, and converges to the expected value given a sufficiently large sample set.

A second property is:
The variance: of the distribution can be described in a certain way.
Variance is a measure of how badly values of samples are spread out from the mean.
In another word: it is the average of the squared differences from the Mean.

Let's get a little bit more formal about five different characteristics
of distributions.

***First, we can talk about the
Distribution central tendency:
When you have two or more numbers it is nice to find a value for the "center".
And the measures we would use for this are mode, median, or mean.
This characteristic is really about where the bulk of probability is in the distribution.

***We can also talk about the
Variability in the distribution:
There are a couple of ways we can speak of this.
The standard deviation is one, the interquartile range is another.
The standard deviation is simply a measure of how different each item, in our sample, is from the mean.
Or it is a measure of how spread out numbers are.

Here's the formula for standard deviation:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

Let's just walk through how we would write this up.
Let's draw 1,000 samples from a normal distribution with an expected value of 0.75 and a standard deviation of 1.
Then we calculate the actual mean using NumPy's mean feature.

The part inside the summation says xi- x bar.
Xi is the current item in the list and x bar is the mean.
So we calculate the difference, then we square the result, then we sum all of these.

This might be a reasonable place to use a map and apply a lambda to calculate the differences between the mean and the measured value.
Then to convert this back to a list, so NumPy can use it.
Now we just have to square each value, sum them together, and take the square root.

```
distribution = np.random.normal(0.75,size=1000)

np.sqrt(np.sum((np.mean(distribution)-distribution)**2)/len(distribution))
```

```
1.0050862291465978
```

So that's the size of our standard deviation. It covers roughly 68% of the area around the mean, split evenly around the side of the mean.
Now we don't normally have to do all this work ourselves, but I wanted to show you how you can sample from the distribution, create a precise programmatic description of a formula, and apply it to your data.

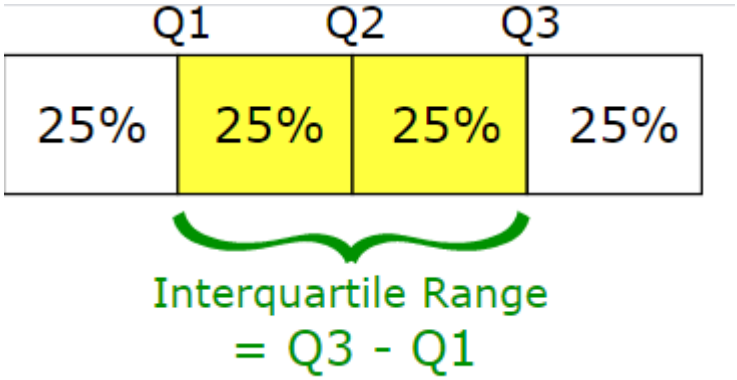
But for standard deviation, which is just one particular measure of variability, NumPy has a built-in function that you can apply, called STD.

```
np.std(distribution)
```

```
1.0050862291465978
```

The interquartile range: is the range from Quartile 1 to Quartile 3:

Q3 − Q1
(Quartiles are the values that divide a list of numbers into quarters.)



There's a couple more measures of distribution that are interesting to talk about.

Kurtosis:
Is the shape of the tails of the distribution.
We can measure the kurtosis using the statistics functions in the SciPy package.

```
import scipy.stats as stats
stats.kurtosis(distribution)
```

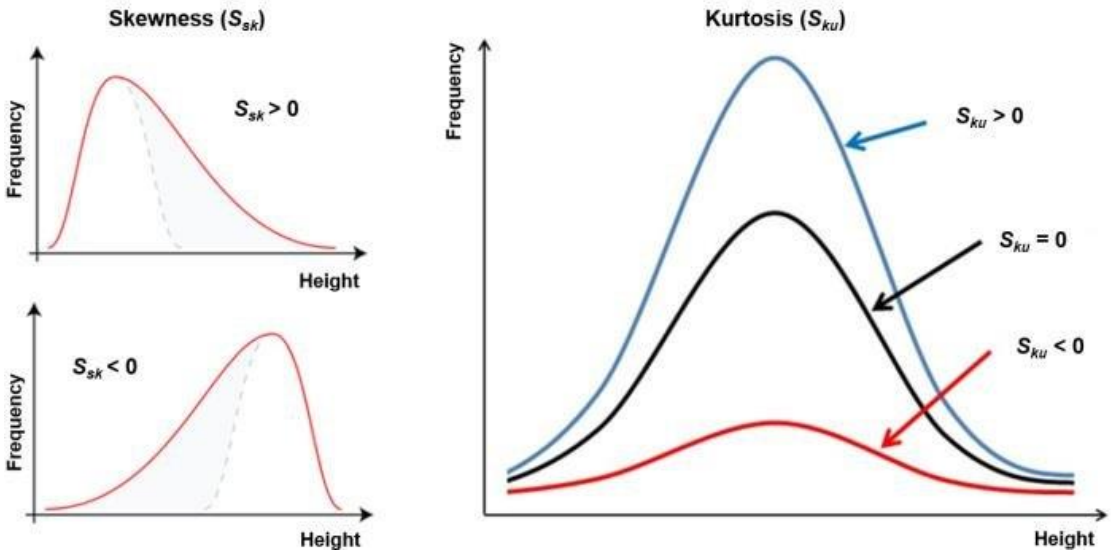
```
0.5190981897849953
```

A negative value means the curve is slightly more flat than a normal distribution, and a positive value means the curve is slightly more peaky than a normal distribution.
Remember that we aren't measuring the kurtosis of the distribution per se, but of the thousand values which we sampled out of the distribution.
This is a sublet but important distinction.

Kurtosis is all about the tails of the distribution — not the peakedness or flatness. It is used to describe the extreme values in one versus the other tail. It is actually the measure of outliers present in the distribution.

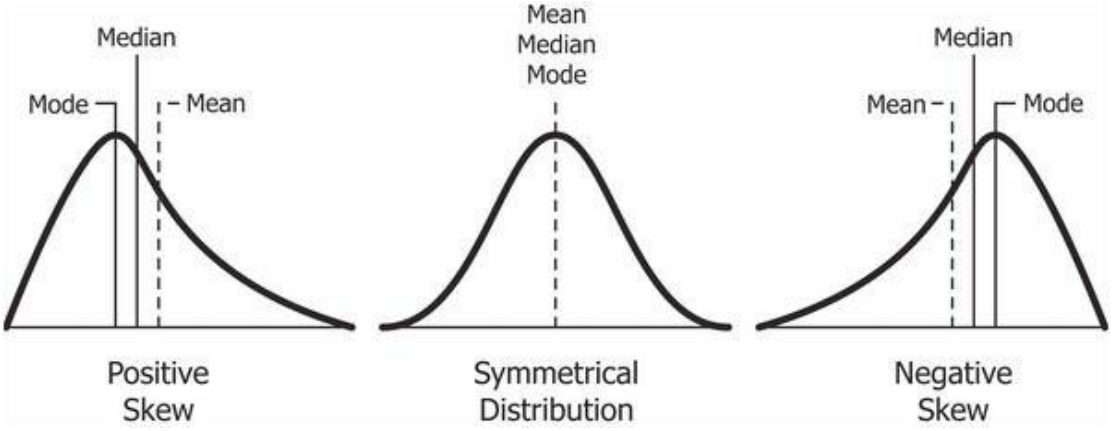
High kurtosis in a data set is an indicator that data has heavy tails or outliers. If there is a high kurtosis, then, we need to investigate why do we have so many outliers. It indicates a lot of things, maybe wrong data entry or other things. Investigate!

Low kurtosis in a data set is an indicator that data has light tails or lack of outliers. If we get low kurtosis(too good to be true), then also we need to investigate and trim the dataset of unwanted results.



Skew:
We could also move out of the normal distributions and push the peak of the curve one way or the other. And this is called the *skew*.

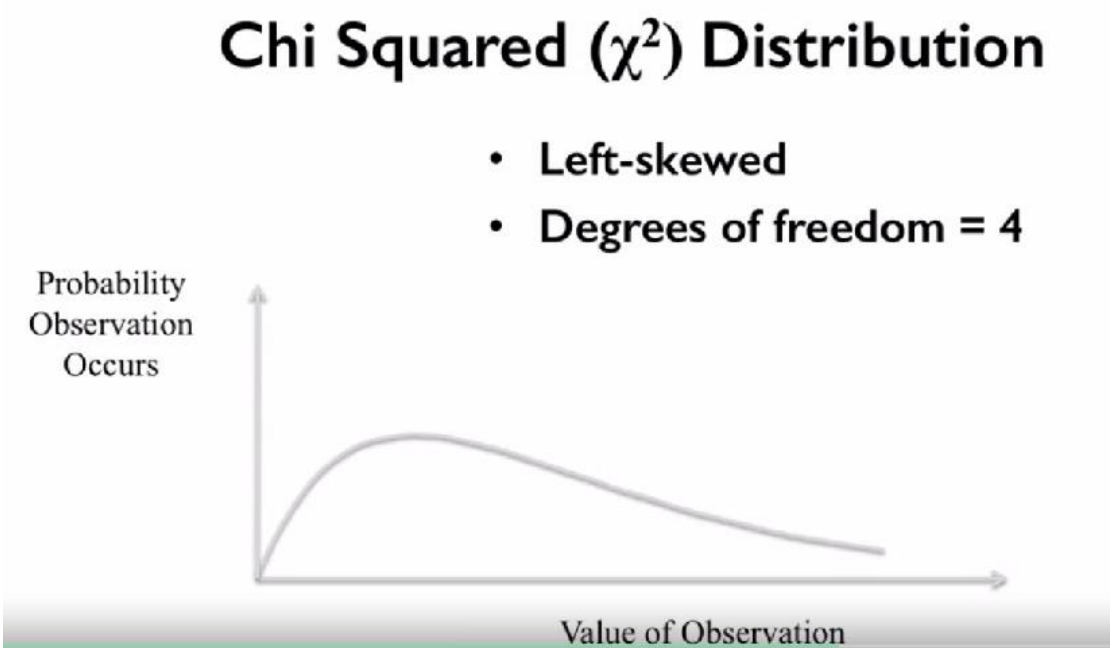
It is the degree of distortion from the symmetrical bell curve or the normal distribution. It measures the lack of symmetry in data distribution.
It differentiates extreme values in one versus the other tail. A symmetrical distribution will have a skewness of 0.
Positive Skewness: means when the tail on the right side of the distribution is longer or fatter. The mean and median will be greater than the mode.
Negative Skewness: is when the tail of the left side of the distribution is longer or fatter than the tail on the right side. The mean and median will be less than the mode.




```
stats.skew(distribution)
```

```
0.047708526631855167
```

If we test our current sample data, we see that there isn't much of a skew.



Let's switch distributions and take a look at a distribution called the Chi Squared distribution, which is also quite commonly used in statistic.

chi-square distribution (also chi-squared or χ2-distribution) with k degrees of freedom is the distribution of a sum of the squares of k independent standard normal random variables.
The Chi Squared Distribution has only one parameter called the degrees of freedom.
The degrees of freedom is closely related to the number of samples that you take from a normal population.
In other words it is the number of values in the final calculation of a statistic that are free to vary
It's important for significance testing.
But what I would like you to observe, is that as the degrees of freedom increases, the shape of the Chi Squared distribution changes.
In particular, the skew to the left begins to move towards the center.
We can observe this through simulation.

First we'll sample 1,000 values from a Chi Squared distribution with degrees of freedom 2.
Now we can see that the skew is quite large.
Now if we re-sample changing degrees of freedom to 5.

We see that the skew has decreased significantly.

```
chi_squared_df2 = np.random.chisquare(2, size=10000)
stats.skew(chi_squared_df2)
```

```
1.9647375786970314
```

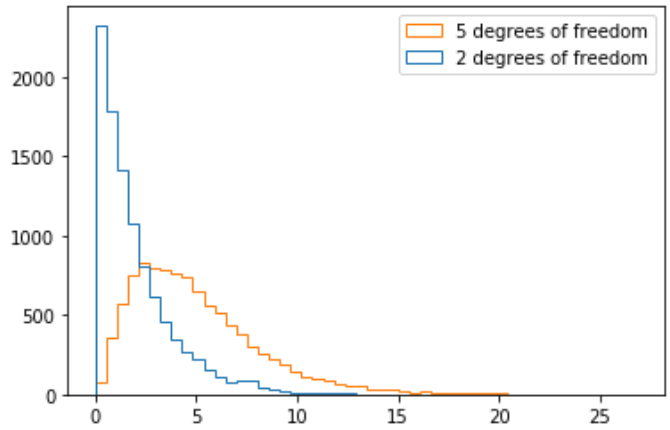
```
chi_squared_df5 = np.random.chisquare(5, size=10000)
stats.skew(chi_squared_df5)
```

```
1.3093023650783777
```

We can actually plot this right in the Jupiter notebook.
You can see a histogram with our plot with the two degrees of freedom is skewed much further to the left, while our plot with the five degrees of freedom is not as highly skewed.

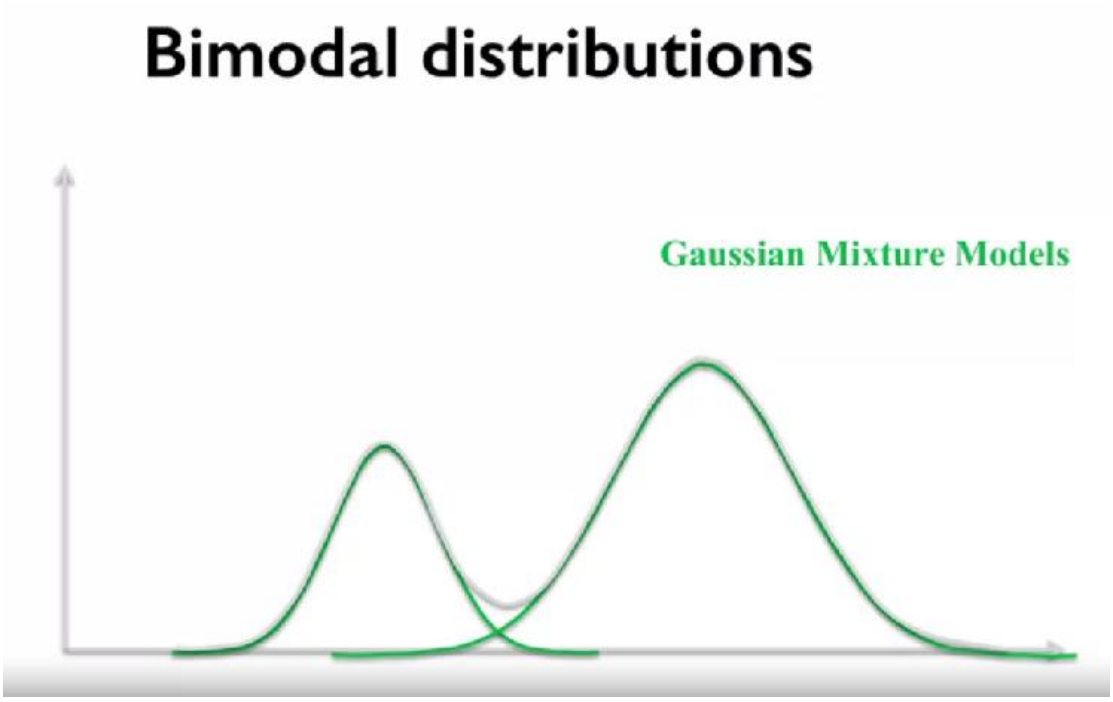
```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

output = plt.hist([chi_squared_df2,chi_squared_df5], bins=50, histtype='step',
                  label=['2 degrees of freedom','5 degrees of freedom'])
plt.legend(loc='upper right')
```

```
<matplotlib.legend.Legend at 0x87b6ec8>
```



I could encourage you as always to play with this notebook and change the parameters and see how the degrees of freedom changes the skew of the distribution.

The last aspect of distributions that I want to talk about is the modality.
So far, all of the distributions I've shown have a single high point, a peak.
But what if we have multiple peaks?
This distribution has two high points, so we call it bimodal.
These are really interesting distributions and happen regularly in data mining.
But a useful insight is that we can actually model these using two normal distributions with different parameters.
These are called Gaussian Mixture Models and are particularly useful when clustering data.



Remember that a distribution is just a shape that describes the probability of a value being pulled when we sample a population.
And NumPy and SciPy each have a number of different distributions built in for us to be able to sample from.

The last point I want to leave you with here is a reference.
If you find this way of exploring statistics interesting. Alan Downey wrote a nice book called Think Stats by the publisher O'Reilly.
I think he does a really nice job of teaching how to think about statistics from a programming perspective, one where you write the functions behind the statistical methods.
It's not really a reference book, but it's an interesting way to approach learning the fundamentals of statistics.
Allen even has a free copy of this book available on his website in PDF format and, of course, all of the code is done in Python.