

Probability

Many of the problems we try to solve using statistics are to do with *probability*. For example, what's the probable salary for a graduate who scored a given score in their final exam at school? Or, what's the likely height of a child given the height of his or her parents?

It therefore makes sense to learn some basic principles of probability as we study statistics.

Probability Basics

Let's start with some basic definitions and principles.

- An **experiment** or **trial** is an action with an uncertain outcome, such as tossing a coin.
- A **sample space** is the set of all possible outcomes of an experiment. In a coin toss, there's a set of two possible outcomes (*heads* and *tails*).
- A **sample point** is a single possible outcome - for example, *heads*
- An **event** is a specific outcome of single instance of an experiment - for example, tossing a coin and getting *tails*.
- **Probability** is a value between 0 and 1 that indicates the likelihood of a particular event, with 0 meaning that the event is impossible, and 1 meaning that the event is inevitable. In general terms, it's calculated like this:

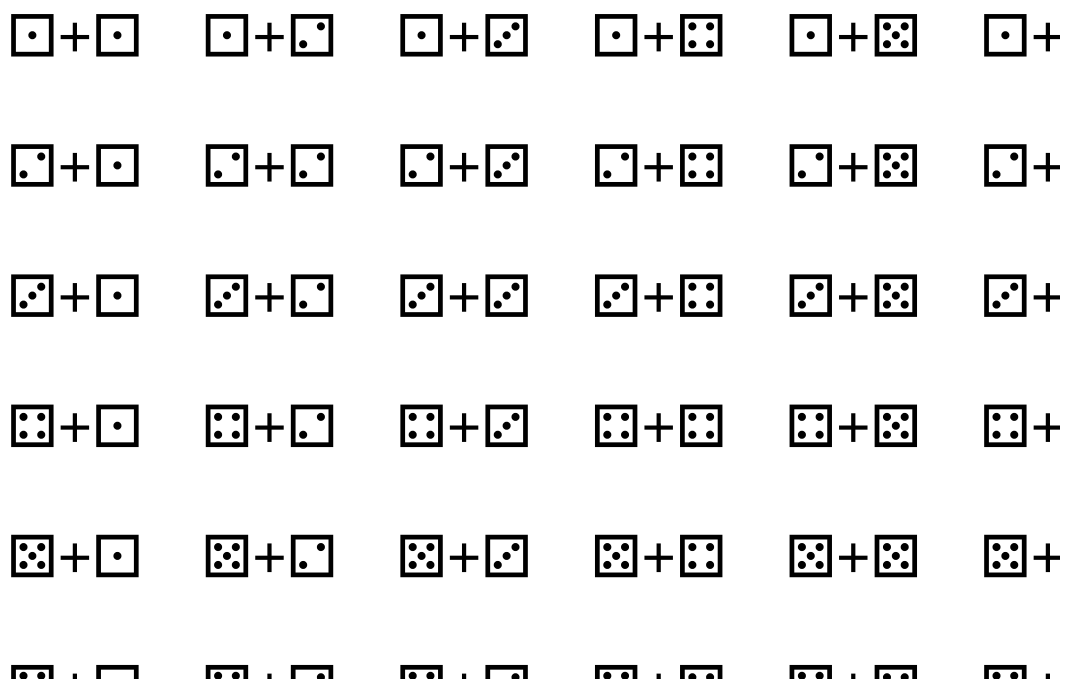
$$\text{probability of an event} = \frac{\text{Number of sample points that produce the event}}{\text{Total number of sample points in the sample space}}$$

For example, the probability of getting *heads* when tossing a coin is $\frac{1}{2}$ - there is only one side of the coin that is designated *heads*. and there are two possible outcomes in the sample space (*heads* and *tails*). So the probability of getting *heads* in a single coin toss is 0.5 (or 50% when expressed as a percentage).

Let's look at another example. Suppose you throw two dice, hoping to get 7.

The dice throw itself is an *experiment* - you don't know the outcome until the dice have landed and settled.

The *sample space* of all possible outcomes is every combination of two dice - 36 *sample points*:





Conditional Probability and Dependence

Events can be:

- *Independent* (events that are not affected by other events)
- *Dependent* (events that are conditional on other events)
- *Mutually Exclusive* (events that can't occur together)

Independent Events

Imagine you toss a coin. The sample space contains two possible outcomes: heads () or tails ().

The probability of getting *heads* is $\frac{1}{2}$, and the probability of getting *tails* is also $\frac{1}{2}$. Let's toss a coin...



OK, so we got *heads*. Now, let's toss the coin again:



It looks like we got *heads* again. If we were to toss the coin a third time, what's the probability that we'd get *heads*?

Although you might be tempted to think that a *tail* is overdue, the fact is that each coin toss is an independent event. The outcome of the first coin toss does not affect the second coin toss (or the third, or any number of other coin tosses). For each independent coin toss, the probability of getting *heads* (or *tails*) remains $\frac{1}{2}$, or 50%.

Run the following Python code to simulate 10,000 coin tosses by assigning a random value of 0 or 1 to *heads* and *tails*. Each time the coin is tossed, the probability of getting *heads* or *tails* is 50%, so you should expect approximately half of the results to be *heads* and half to be *tails* (it won't be exactly half, due to a little random variation; but it should be close):

```
In [1]: %matplotlib inline
import random

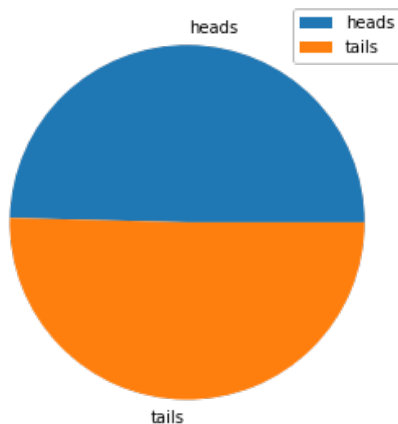
# Create a list with 2 element (for heads and tails)
heads_tails = [0,0]

# loop through 10000 trials
trials = 10000
trial = 0
while trial < trials:
    trial = trial + 1
    # Get a random 0 or 1
    toss = random.randint(0,1)
    # Increment the list element corresponding to the toss result
    heads_tails[toss] = heads_tails[toss] + 1

print (heads_tails)

# Show a pie chart of the results
from matplotlib import pyplot as plt
plt.figure(figsize=(5,5))
plt.pie(heads_tails, labels=['heads', 'tails'])
plt.legend()
plt.show()

[4960, 5040]
```



Combining Independent Events

Now, let's ask a slightly different question. What is the probability of getting three *heads* in a row? Since the probability of a heads on each independent toss is $\frac{1}{2}$, you might be tempted to think that the same probability applies to getting three *heads* in a row; but actually, we need to treat getting three *heads* as it's own event, which is the combination of three independent events. To combine independent events like this, we need to multiply the probability of each event. So:

$$\text{☀} = \frac{1}{2}$$

$$\text{☀} \text{ ☀} = \frac{1}{2} \times \frac{1}{2}$$

$$\text{☀} \text{ ☀} \text{ ☀} = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2}$$

So the probability of tossing three *heads* in a row is $0.5 \times 0.5 \times 0.5$, which is 0.125 (or 12.5%).

Run the code below to simulate 10,000 trials of flipping a coin three times:

```
# Count the number of 3xHeads results
h3 = 0

# Create a list of all results
results = []

# loop through 10000 trials
trials = 10000
trial = 0
while trial < trials:
    trial = trial + 1
    # Flip three coins
    result = ['H' if random.randint(0,1) == 1 else 'T',
              'H' if random.randint(0,1) == 1 else 'T',
              'H' if random.randint(0,1) == 1 else 'T']
    results.append(result)
    # If it's three heads, add it to the count
    h3 = h3 + int(result == ['H', 'H', 'H'])

# What proportion of trials produced 3x heads
print("%.2f%%" % ((h3/trials)*100))

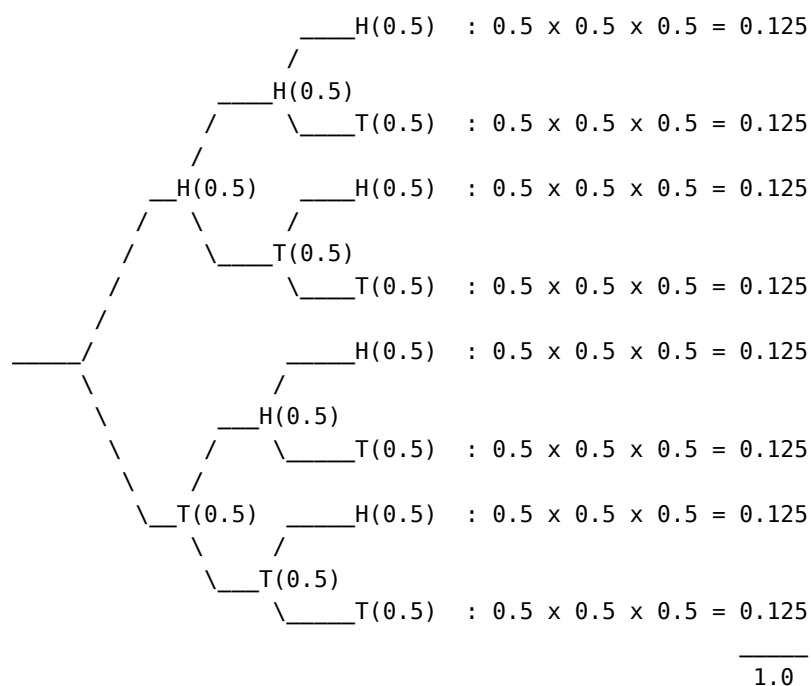
# Show all the results
print(results)
```

[illegible]

The output shows the percentage of times a trial resulted in three heads (which should be somewhere close to 12.5%). You can count the number of *['H', 'H', 'H']* entries in the full list of results to verify this if you like!

Probability Trees

You can represent a series of events and their probabilities as a probability tree:



Starting at the left, you can follow the branches in the tree that represent each event (in this case a coin toss result of *heads* or *tails* at each branch). Multiplying the probability of each branch of your path through the tree gives you the combined probability for an event composed of all of the events in the path. In this case, you can see from the tree that you are equally likely to get any sequence of three *heads* or *tails* results (so three *heads* is just as likely as three *tails*, which is just as likely as *head-tail-head*, *tail-head-tail*, or any other combination!)

Note that the total probability for all paths through the tree adds up to 1.

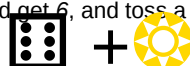
Combined Event Probability Notation

When calculating the probability of combined events, we assign a letter such as **A** or **B** to each event, and we use the *intersection* (\cap) symbol to indicate that we want the combined probability of multiple events. So we could assign the letters **A**, **B**, and **C** to each independent coin toss in our sequence of three tosses, and express the combined probability like this:

$$P(A \cap B \cap C) = P(A) \times P(B) \times P(C)$$

Combining Events with Different Probabilities

Imagine you have created a new game that mixes the excitement of coin-tossing with the thrill of die-rolling! The objective of the game is to roll a die and get 6, and toss a coin and get *heads*:



On each turn of the game, a player rolls the die and tosses the coin.

How can we calculate the probability of winning?

There are two independent events required to win: a die-roll of 6 (which we'll call event **A**), and a coin-toss of *heads* (which we'll call event **B**)

Dependent Events

Let's return to our deck of 52 cards from which we're going to draw one card. The sample space can be summarized like this:

13 x  13 x  13 x  13 x 

There are two black suits (*spades* and *clubs*) and two red suits (*hearts* and *diamonds*); with 13 cards in each suit. So the probability of drawing a black card (event **A**) and the probability of drawing a red card (event **B**) can be calculated like this:

$$P(A) = \frac{13 + 13}{52} = \frac{26}{52} = 0.5 \quad P(B) = \frac{13 + 13}{52} = \frac{26}{52} = 0.5$$

Now let's draw a card from the deck:



We drew a heart, which is red. So, assuming we don't replace the card back into the deck, this changes the sample space as follows:

12 x  13 x  13 x  13 x 

The probabilities for **A** and **B** are now:

$$P(A) = \frac{13 + 13}{51} = \frac{26}{51} = 0.51 \quad P(B) = \frac{12 + 13}{51} = \frac{25}{51} = 0.49$$

Now let's draw a second card:



We drew a diamond, so again this changes the sample space for the next draw:

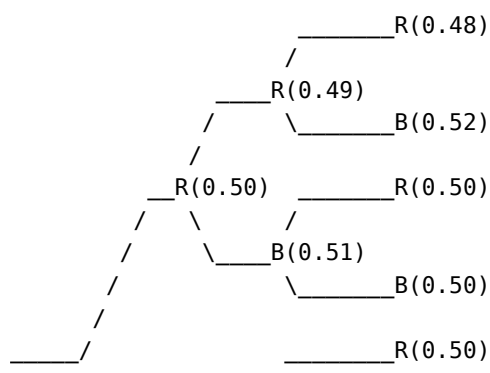
12 x  13 x  13 x  12 x 

The probabilities for **A** and **B** are now:

$$P(A) = \frac{13 + 13}{50} = \frac{26}{50} = 0.52 \quad P(B) = \frac{12 + 12}{50} = \frac{24}{50} = 0.48$$

So it's clear that one event can affect another; in this case, the probability of drawing a card of a particular color on the second draw depends on the color of card drawn on the previous draw. We call these *dependent* events.

Probability trees are particularly useful when looking at dependent events. Here's a probability tree for drawing red or black cards as the first three draws from a deck of cards:



Binomial Variables and Distributions

Now that we know something about probability, let's apply that to statistics. Statistics is about inferring measures for a full population based on samples, allowing for random variation; so we're going to have to consider the idea of a *random variable*.

A random variable is a number that can vary in value. For example, the temperature on a given day, or the number of students taking a class.

Binomial Variables

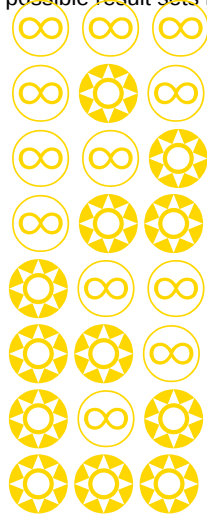
One particular type of random variable that we use in statistics is a *binomial* variable. A binomial variable is used to count how frequently an event occurs in a fixed number of repeated independent experiments. The event in question must have the same probability of occurring in each experiment, and indicates the success or failure of the experiment; with a probability p of success, which has a complement of $1 - p$ as the probability of failure (we often call this kind of experiment a *Bernoulli Trial* after Swiss mathematician Jacob Bernoulli).

For example, suppose we flip a coin three times, counting *heads* as success. We can define a binomial variable to represent the number of successful coin flips (that is, the number of times we got *heads*).

Let's examine this in more detail.

We'll call our variable X , and as stated previously it represents the number of times we flip *heads* in a series of three coin flips. Let's start by examining all the possible values for X .

We're flipping the coin three times, with a probability of $1/2$ of success on each flip. The possible results include none of the flips resulting in *heads*, all of the flips resulting in *heads*, or any combination in between. There are two possible outcomes from each flip, and there are three flips, so the total number of possible result sets is 2^3 , which is 8. Here they are:



In these results, our variable X , representing the number of successful events (getting *heads*), can vary from 0 to 3. We can write that like this:

$$X = \{0, 1, 2, 3\}$$

When we want to indicate a specific outcome for a random variable, we use write the variable in lower case, for example x . So what's the probability that $x = 0$ (meaning that out of our three flips we got no *heads*)?

We can easily see, that there is 1 row in our set of possible outcomes that contains no *heads*, so:

$$P(x = 0) = \frac{1}{8}$$

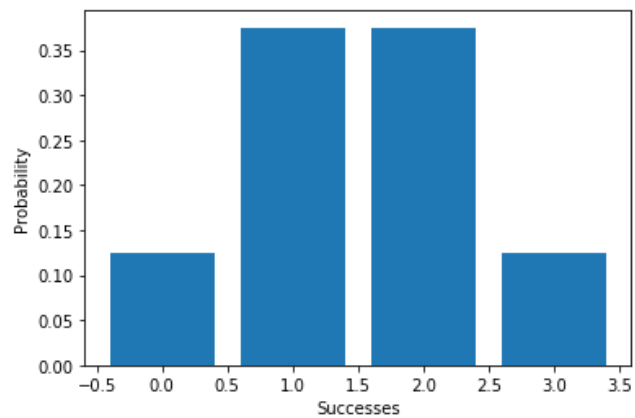
```
In [3]: %matplotlib inline
from scipy import special as sps
from matplotlib import pyplot as plt
import numpy as np

trials = 3

possibilities = 2**trials
x = np.array(range(0, trials+1))

p = np.array([sps.comb(trials, i, exact=True)/possibilities for i in x])

# Set up the graph
plt.xlabel('Successes')
plt.ylabel('Probability')
plt.bar(x, p)
plt.show()
```



Allowing for Bias

Previously, we calculated the probability for each possible value of a random variable by simply dividing the number of combinations for that value by the total number of possible outcomes. This works if the probability of the event being tested is equal for failure and success; but of course, not all experiments have an equal chance of success or failure. Some include a bias that makes success more or less likely - so we need to be a little more thorough in our calculations to allow for this.

Suppose you're flying off to some exotic destination, and you know that there's a one in four chance that the airport security scanner will trigger a random search for each passenger that goes through. If you watch five passengers go through the scanner, how many will be stopped for a random search?

It's tempting to think that there's a one in four chance, so a quarter of the passengers will be stopped; but remember that the searches are triggered randomly for thousands of passengers that pass through the airport each day. It's possible that none of the next five passengers will be searched; all five of them will be searched, or some other value in between will be searched.

Even though the probabilities of being searched or not searched are not the same, this is still a binomial variable. There are a fixed number of independent experiments (five passengers passing through the security scanner), the outcome of each experiment is either success (a search is triggered) or failure (no search is triggered), and the probability of being searched does not change for each passenger.

There are five experiments in which a passenger goes through the security scanner, let's call this **n**.

For each passenger, the probability of being searched is $\frac{1}{4}$ or 0.25. We'll call this **p**.

The complement of **p** (in other words, the probability of *not* being searched) is **1-p**, in this case $\frac{3}{4}$ or 0.75.

So, what's the probability that out of our **n** experiments, three result in a search (let's call that **k**) and the remaining ones (there will be **n-k** of them, which is two) don't?

- The probability of three passengers being searched is $0.25 \times 0.25 \times 0.25$ which is the same as 0.25^3 . Using our generic variables, this is **p^k**.
- The probability that the rest don't get searched is 0.75×0.75 , or 0.75^2 . In terms of our variables, this is **1-p^(n-k)**.
- The combined probability of three searches and two non-searches is therefore $0.25^3 \times 0.75^2$ (approximately 0.088). Using our variables, this is:

$$p^k (1 - p)^{(n-k)}$$

This formula enables us to calculate the probability for a single combination of **n** passengers in which **k** experiments had a successful outcome. In this case, it enables us to calculate that the probability of three passengers out of five being searched is approximately 0.088. However, we need to consider that there are multiple ways this can happen. The first three passengers could get searched; or the last three; or the first, third, and fifth, or any other possible combination of 3 from 5.

There are two possible outcomes for each experiment; so the total number of possible combinations of five passengers being searched or not searched is 2^5 or 32. So within those 32 sets of possible result combinations, how many have three searches? We can use the ${}_nC_k$ formula to calculate this:

$${}_5C_3 = \frac{5!}{3!(5-3)!} = \frac{120}{6 \times 4} = \frac{120}{24} = 5$$

So 5 out of our 32 combinations had 3 searches and 2 non-searches.

To find the probability of any combination of 3 searches out of 5 passengers, we need to multiply the number of possible combinations by the probability for a single combination - in this case $\frac{5}{32} \times 0.088$, which is 0.01375, or 13.75%.

So our complete formula to calculate the probability of **k** events from **n** experiments with probability **p** is:

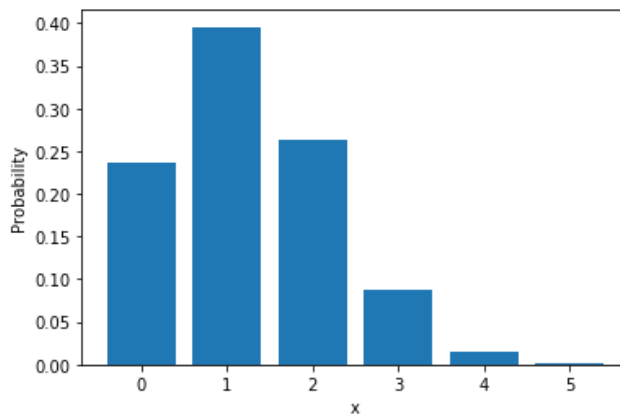
$$P(X = k) = \frac{n!}{k!(n-k)!} p^k (1-p)^{(n-k)}$$

```
In [4]: %matplotlib inline
from scipy.stats import binom
from matplotlib import pyplot as plt
import numpy as np

n = 5
p = 0.25
x = np.array(range(0, n+1))

prob = np.array([binom.pmf(k, n, p) for k in x])

# Set up the graph
plt.xlabel('x')
plt.ylabel('Probability')
plt.bar(x, prob)
plt.show()
```



You can see from the bar chart that with this small value for n , the distribution is right-skewed.

Recall that in our coin flipping experiment, when the probability of failure vs success was equal, the resulting distribution was symmetrical. With an unequal probability of success in each experiment, the bias has the effect of skewing the overall probability mass.

However, try increasing the value of n in the code above to 10, 20, and 50; re-running the cell each time. With more observations, the *central limit theorem* starts to take effect and the distribution starts to look more symmetrical - with enough observations it starts to look like a *normal* distribution.

There is an important distinction here - the *normal* distribution applies to *continuous* variables, while the *binomial* distribution applies to *discrete* variables. However, the similarities help in a number of statistical contexts where the number of observations (experiments) is large enough for the *central limit theorem* to make the distribution of binomial variable values behave like a *normal* distribution.

Working with the Binomial Distribution

Now that you know how to work out a binomial distribution for a repeated experiment, it's time to take a look at some statistics that will help us quantify some aspects of probability.

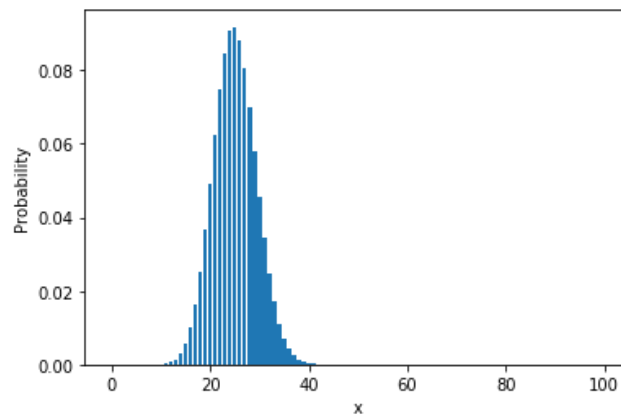
Let's increase our n value to 100 so that we're looking at the number of searches per 100 passengers. This gives us the binomial distribution graphed by the following code:

```
In [5]: %matplotlib inline
from scipy.stats import binom
from matplotlib import pyplot as plt
import numpy as np

n = 100
p = 0.25
x = np.array(range(0, n+1))

prob = np.array([binom.pmf(k, n, p) for k in x])

# Set up the graph
plt.xlabel('x')
plt.ylabel('Probability')
plt.bar(x, prob)
plt.show()
```



Mean (Expected Value)

We can calculate the mean of the distribution like this:

$$\mu = np$$

So for our airport passengers, this is:

$$\mu = 100 \times 0.25 = 25$$

When we're talking about a probability distribution, the mean is usually referred to as the *expected value*. In this case, for any 100 passengers we can reasonably expect 25 of them to be searched.

Variance and Standard Deviation

Obviously, we can't search a quarter of a passenger - the expected value reflects the fact that there is variation, and indicates an average value for our binomial random variable. To get an indication of how much variability there actually is in this scenario, we can calculate the variance and standard deviation.

For variance of a binomial probability distribution, we can use this formula:

$$\sigma^2 = np(1 - p)$$

So for our airport passengers:

$$\sigma^2 = 100 \times 0.25 \times 0.75 = 18.75$$

To convert this to standard deviation we just take the square root:

$$\sigma = \sqrt{np(1 - p)}$$

So:

$$\sigma = \sqrt{18.75} \approx 4.33$$

So for every 100 passengers, we can expect 25 searches with a standard deviation of 4.33

In Python, you can use the **mean**, **var**, and **std** functions from the `scipy.stats.binom` package to return binomial distribution statistics for given values of *n* and *p*:

In [6]: `from scipy.stats import binom`

```
n = 100
p = 0.25

print(binom.mean(n,p))
print(binom.var(n,p))
print(binom.std(n,p))

25.0
18.75
4.330127018922194
```