GRESHAM COLLEGE

9th January 2018

MATHS IS CODED IN YOUR GENES

PROFESSOR CHRISTOPHER BUDD

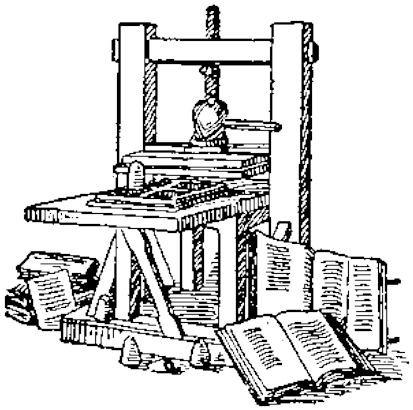
Introduction

We are surrounded by information and are constantly receiving and transmitting it to other people all over the world. With good reason we can call the 21st Century the *Information Age*.

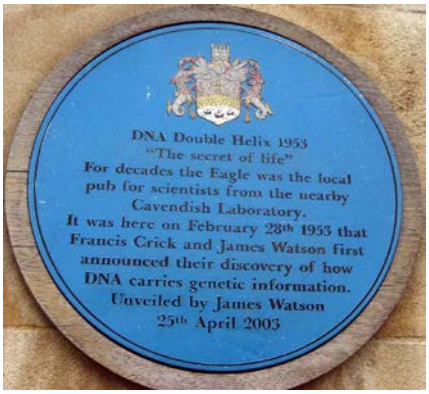
When using the Internet, we meet terms such as byte, MP3, JPEG and DSTL without (I suspect in most cases) understanding what they mean. All of these are important technologies arising from the theory of information, which lies at the heart of this lecture.

Of course, the transmission, and the reception, of information is nothing new. A great leap forward in human civilisation came when we learned how to speak and could convey information using words. Our ability to record and transmit information changed again with the development of the first writing systems. Arguably using clay tablets with the Babylonians. Below we can see a clay tablet from Sumeria, Mesopotamia, with early mathematics recorded on it in the ancient cuneiform script. This can now be seen in the British Museum.

Much later in the 15th Century both the *speed* and the *reliability* of written communication was transformed by the invention of the printing press by Johannes Gutenberg. His original press is illustrated below.



This invention allowed the rapid spread of ideas and arguably was one of the driving forces behind the Reformation. We are now seeing a revolution every bit as big as that of the invention of the printing press. The modern use of computers, Wi-Fi, the Internet, Satellites and mobile phones now give us the ability to transmit huge amounts of information reliably over vast distances. Furthermore, technologies such as JPEG and MP3 players owe their success to the ability to compress useful information into a small amount of memory without any significant loss of content. Where this will lead us is most unclear! However, nature has been transmitting huge quantities of information reliably well before humans did. Indeed, the fact that animals can reproduce reliably to create copies of their species is clear evidence of this. Darwin understood this, but the reason behind was not uncovered until Mendel discovered the nature of genetics in 1866. Then in 1953, Franklin, Watson and Crick unravelled the secrets of DNA, and revealed their discovery in the Eagle Pub in Cambridge. This is recorded on the following plaque on the pub wall



This discovery more than anything showed not only how nature transmitted the information of inheritance, but also how it could make errors in transmission, leading to the evolution of the species.

In this talk I will describe the nature of information, and how it is transmitted, both by humans and by machines, with as little error as possible, and corrected if mistakes have been in the transmission. We will then look at how human transmission compares with that used by nature and how this relates to evolution.

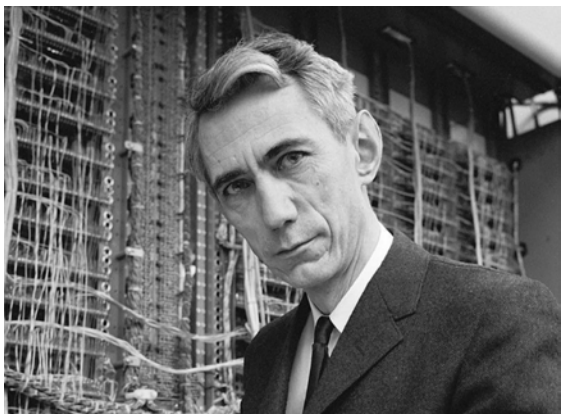
The Nature of Information

We all have an intuitive idea of what passing information is. Essentially it is the transmission of something useful from the sender to the receiver. But what do we mean by useful? We like to distinguish between the *useful content* of a signal and that part of the signal which is not useful and which we usually call *noise*. In the context of genetics useful information comprises the chromosomes in the DNA molecule, which allow the creation of a

new, and healthy, individual. However, it is not always easy to distinguish between a useful signal and noise. To give an example, suppose that I am making a phone call. Whilst on the phone, the words of the speaker are useful content. I then hear a car starting up in the background. As far as my conversation on the phone is concerned this is noise. It makes it slightly harder to hear the person on the other end of the line, but I can still make out their conversation. This state persists until I realise that the car noise is in fact that of my own car, which is in the process of being stolen! This illustrates the difficulty in separating the noise in a signal from its content. However, this is exactly what we need to do every time we receive a signal, whether it is a picture transmitted from a distant satellite or the information in a DNA molecule.

Originally, we transmitted information by word of mouth and then by the written word. This was slow, and whilst mistakes could be (and were) made in transmitting this, they could generally be corrected almost immediately. However, this all changed with the advent of electronic communication starting with the telegraph in the 19th century and then wireless transmission. The study of the nature information became vital when the transmission rates became too high to allow for a quick correction, and the distance of transmission led to much noisier signals. Furthermore, the sheer size of modern data means that it is impossible to send useful content without some form of *data compression.* The study and understanding of information lies at the heart of much of modern technology including satellites, Smart and mobile phones, the Internet, CDs and MP3 players, to name just a few.

Perhaps the first person to study the nature of information systematically was Claude Shannon, pictured below, and often called the father of information. Shannon was working at the Bell Labs in the USA and in 1948 published a seminal paper called *A mathematical theory of communication* [1]. This paper introduced the subject of information theory which according to Wikipedia [2] is: *the study of the transmission, processing, extraction, and utilization of information.*



In his paper Shannon studied the storage and the communication of information as well as methods of coding it. Shannon considered passing information through a channel (which could for example, be a radio link, a phone line or a connection to a satellite). Typically, such information is sent through the channel by using binary digits or bits, and we will look at this in more detail in the next section. Provided that the channel has been well designed, most of the information will be transmitted correctly. However, some of the bits in the message will be corrupted. For example, we try to send a 1, and in the process of sending someone turns on a switch, and the resulting electrical pulse turns the 1 into a 0. The longer the channel (for example a distant satellite transmitting a picture of a planet to Earth) the more likely this is to happen. We call this the channel noise. There will be a probability p of a bit being corrupted by noise and this will lead to errors in the signal. To be able to resolve the information in the signal we must somehow remove the uncertainty introduced by the random channel noise. Shannon considered a situation where one person transmitted the information through the channel. The information was then corrupted by noise and received at the other end by a receiver. The challenge of the receiver is then to reconstruct the original message as well as possible despite the errors due to noise. The big question is how well this can be done, and how much noise can we cope with before it becomes impossible? Or to quote Shannon directly

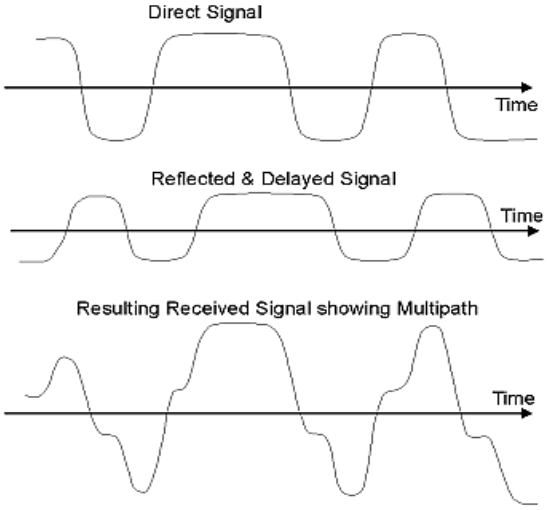
The fundamental problem of communication is that of reproducing at one point, either exactly or approximately, a message selected at another point."

In order to answer this question Shannon had to understand how much information there was in a signal and how certain or uncertain it is that the information is received correctly when it is subject to a random variation. Shannon's theorem, which he described in [1], then expressed the maximum information rate at which reliable communication is possible over a channel that has a certain error probability or signal to noise ratio (SNR). This was called the channel capacity. Shannon's theorem is the fundamental basis of nearly all modern communication, and much effort has been put into the design of codes which allow Shannon's upper bound to be achieved. We will meet these codes presently. To quote Wikipedia [2]

Its impact has been crucial to the success of the Voyager missions to deep space, the invention of the compact disc, the feasibility of mobile phones, the development of the Internet, the study of linguistics and of human perception, the understanding of black holes, and numerous other fields.

It is worth pointing out that there are many other, and more systematic, ways that the

information in a signal can be corrupted, particularly one such a radio signal which is transmitted over a long distance. During this transmission process some parts of the signal will travel faster than others, some will be delayed, some will be blurred, some will be reflected, and some will go on multiple paths. The received signal will then often differ quite a lot from the signal, which has been transmitted. An example of this form of distorti0n is shown below.



Thus, we receive a signal which has been distorted both by the systematics processes above, and also by the effects of random noise. Mathematically we can say that if u(t) is the transmitted signal and v(t) is the received signal, then

$$v(t) = \sum_{i=1}^{N} f_i(t) * u(t) + e(t)$$

Where * is the convolution operation, and this formula assumes that the signal has gone through N different paths before being received. In this formula e(t) represents the addition of random noise to the signal. In the earlier lecture on 'How maths can save your life' I talked about how the amount of blurring in a signal can be reduced. We will now look at how the signal u(t) can be *encoded* so that *despite* the corruption to the signal above, and the addition of noise, the original information can be recovered.

Digital Data Transmission

There are basically two types of signal, which are called *analogue* and *digital*. Until recently TV signals were transmitted as analogue signals and we have recently see a change from this to digital transmission. An *analogue signal* can take a continuous range of values. For example, the field strength of a radio signal (measured in micro volts per metre), the sound strength of an audio signal measured in decibels (such as human speech), or the collar length of a made to measure suit. In contrast *digital signals* take values over a much more limited range of values, often just 0 and 1. An analogy to this is buying a shirt which comes in a range of distinct sizes such as 10-16. This process reduce the content of a signal to a series of 1s and 0s so that a message might read …00110100110010011100… . The advantage of this over other types of transmission (such as human speech) is that the difference between a 1 and a 0 is unambiguous, whilst the difference between two analogue values, say 0.48 and 0.49 can be small. Therefore, a random error is much more likely to affect an analogue signal than a digital one. It is also much easier to store a 1 or a zero in a computer memory than an analogue value. For example, 0 can be the off state of a (flip-flop/bi-stable) circuit, and a 1 the on state. Equally well it could be the on state of a relay, or even a hole in a punch card or paper tape.

The idea of representing information in the form of numbers, in terms of 0s and 1s is due to

Leibnitz in 1679. He invented (discovered?) binary arithmetic (although there is, as ever, evidence that the Chinese invented it much earlier, and that Leibnitz was aware of their work). This can be found in *Explication de l'Arithmétique Binaire* [3] which was published in 1703. In the binary system numbers are expressed in Base 2. In this system

0 0 0 1 has the numerical value of $1 = 2^0$

0 0 1 0 has the numerical value of $2 = 2^1$

0 1 0 0 has the numerical value of $4 = 2^2$

1 0 0 0 has the numerical value of $8 = 2^3$

The first 16 decimal numbers and their binary equivalents are as follows

0 0 0 0	0	1 0 0 0	8
0 0 0 1	1	1 0 0 1	9
0 0 1 0	2	1 0 1 0	10
0 0 1 1	3	1 0 1 1	11
0 1 0 0	4	1 1 0 0	12
0 1 0 1	5	1 1 0 1	13
0 1 1 0	6	1 1 1 0	14
0 1 1 1	7	1 1 1 1	15

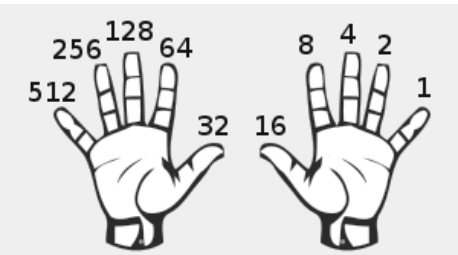
As an example of the ability of binary numbers to store information consider the following joke.

How does a monster count to 30?

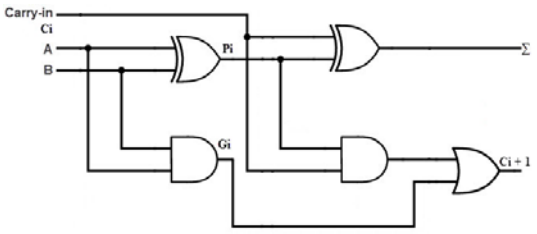
The answer is of course

On their fingers!

In fact, we can all count to 30 on our fingers. In fact, we can count to 31 with the fingers of *one hand*. To do this you use your fingers to represent binary digits, so that an extended finger means a binary digit 1 and a folded finger a zero. The place value of each of the ten fingers is indicated below, allowing you to *count to 1023 using both hands*.



Arithmetic using binary numbers is very simple and can be done very quickly with computer circuits called binary adders. See below for an example. These are usually composed of simple logic gates such as AND and OR gates.



These gates implement, at a circuit level, a development of binary numbers called Boolean Algebra, which was invented by the Irish mathematician, George Boole in 1854. The credit for the first such circuit is due to Shannon (remember him) in his master's thesis in 1937 that implemented Boolean algebra and binary arithmetic using electronic relays and switches for the first time, and founded modern digital circuit design

A single binary digit is called a BIT and carries limited information. Usually the 1s and 0s are grouped together to make up symbols. As an example, the very commonly used ASCII code (American Standard Code for Information Interchange) encodes the letters and punctuation in eight bits. A group of such 8 bits is called a BYTE and a byte can represent 256 (2 to the power of 8) pieces of information. The ASCII code for the letters and numbers is given below.

ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

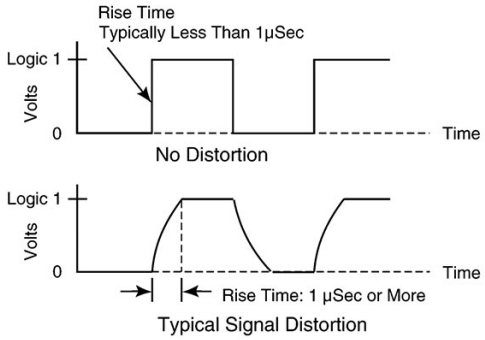
For example, the word *Gresham* would have the ASCII code

0100 0111 0111 0010 0110 0101 0111 0011 0110 1000 0110 0001 0110 1101

Whilst ASCII is adequate for letters in English, in order to encode the whole range of characters found in other languages, and also different symbols, it is now common to use Unicode which encodes a symbol in up to 32 bits.

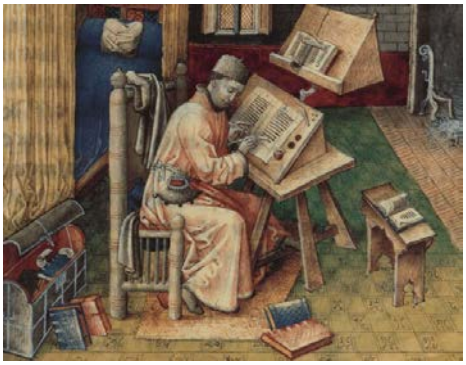
In more sophisticated forms of digital transmission, signals such as music are encoded in binary. For example, **mp3** is a coding format for digital audio devices such as iPods.

One of the big advantages of digital transmission of information using binary coding is that it reduces the error due to spreading/blurring and multi-path delay. As an example, below, we can see a bit as transmitted and then as received.



Whilst distorted it is still recognisable as a bit, and provided that the spread does not extend to the next bit, it is very likely that the original bits from the signal can be recovered.

However, despite the best of our efforts, there is still the possibility that an error will be made and a 1 will be recognised as a 0 or vice versa. If this occurs, then the information in the message will be corrupted. If this happens, it is important that the message is recognised as being wrong to prevent false information being conveyed. The need for the detection of errors has been recognised since the earliest scribes copied manuscripts by hand.

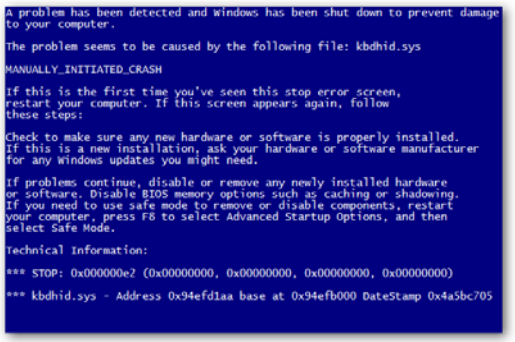


It was important to copy these without error. But to check every word would have been too large a task. Instead various checks were used. For example, when copying the Torah he letters, words, and paragraphs had to be counted, and the middle paragraph, word and letter had to correspond to those of the original document. Similarly, when transmitting binary information, a simple check is used often by the use of a *hash function,* which adds a fixed-length *tag* to a message. This allows the receiver to verify the delivered message by re computing the tag and comparing it with the one provided in the message. A simple example of such a tag comes from including a *check* or *parity* digit to each block of data. This is an extra piece of information (often called redundancy in the message) that is included to check that the message has been transmitted correctly. The simplest example of a check digit is to add up the digits, and then to add an extra digit to make sure that you get an even number. So, for example, if the original transmission was 111 then an extra 1 as the check/parity digit would be added so that 1111 is sent. Alternatively, if the transmission was 101 then the check 010 would be 0 and 0100 would be sent. On receiving the transmission, the computer would add up the digits, and if the sum was not even then it would record an error. An example of the use of this technology can be found on the **bar codes**, which are used on nearly all mass-produced consumer goods sold. Such consumer products typically use either UPC-A or EAN-13 codes. EAN-13 describes a 13-digit sequence, which is broken into four groups. The first two digits of the bar code describe the *number system*. This can either indicate the nationality of the manufacturer, or describe one of a few other categories, such as the ISBN (book identifier) numbers. The next five digits are a manufacturer's identification. The five digits that follow are a product identification number, assigned by the manufacturer. The last digit is a *parity digit*, allowing a scanner to validate whether the barcode has been read correctly.

Similar check digits are used in the numbers on your credit card, which are usually a sequence of decimal digits. In this case the *Luhn Algorithm* is used, in which starting from the right of the credit card number, the digits are alternatively doubled. All of the digits are then added up, and the credit card number passes if the resulting sum is divisible by 10. This process checks both the digits and also the order of the digits in the credit card.

Error Correcting Codes

We have seen how to transmit a message in an unambiguous way, and even to check whether it has been sent with an error or not. However, suppose that we have detected that an error has been made. How can we proceed to find the information in the message? There are various approaches to this. One is simply to (effectively) *panic*, to shut the whole system down and not to proceed until the problem has been fixed. The infamous *blue screen of death* pictured below, and familiar to many computer operators, is an example of this. We show this below



The rationale behind this approach is that in some (indeed many) cases it is better to do nothing than to do something which you know is wrong. However, all we can do at this point is to start again from scratch, and we lose all information in the process.

A second approach called the Automatic Repeat Request (ARQ), which is often used on the Internet, is for the message to be repeated if it is thought to contain an error. We do this all the time. For example, if you scan in an item at the supermarket and the scanner does not recognise it, then you simply scan it in again.

However, this option is not available to us if we are receiving information from, say, a satellite, a mobile phone or from a CD. In this case if we know that an error has been made in transmission then we have to attempt to *correct* it. The general idea for achieving correction is to add some extra data to a message, which allows us to recover it even after an error has been made. To do this, the transmitter sends the original data, and attaches a fixed number of *check bits* using an Error Correcting Code (ECC). The idea behind this is to make the symbols from the different characters in the code as different from each other as possible, so that even if one symbol in the code was corrupted by noise it could still be distinguished from other symbols in the code. Error correcting codes were invented in 1947, at Bell Labs, by the American mathematician Richard Hamming, pictured below.

Error correcting codes put into practice a consequence of Shannon's theorem, which said that there existed codes for which the probability of error on a channel could be made arbitrarily small by increasing the encoding length (that is the amount of redundancy in the symbols making up the code). All of this was provided that the code rate was smaller than the channel capacity. Or in other words, if you were prepared to send information slower and with more redundant (or extra) data, then you could get the message through, even though there was a lot of interference on the way. This result is vital to satellite transmission over vast distances and with huge interference from the Earth's atmosphere. It can certainly be said that it was the invention of Error Correction that made the space race possible. In a sense, this idea is not a new one. Morse code was invented by Samuel Morse in the 19th Century, as a way of encoding letters in such a way that they could be transmitted long distances. Even after the invention of telephony, or the transmission of human speech, Morse code, or telegraphy. Was still used when the channel was noisy. This was because Morse Code characters (illustrated below alongside a Morse Key) were easier to separate than the sounds of the spoken word. Error correcting codes do the same but in a much more sophisticated manner.

A ●−	J ●−−−	S ●●●
B −●●●	K −●−	T −
C −●−●	L ●−●●	U ●●−
D −●●	M −−	V ●●●−
E ●	N −●	W ●−−
F ●●−●	O −−−	X −●●−
G −−●	P ●−−●	Y −●−−
H ●●●●	Q −−●−	Z −−●●
I ●●	R ●−●	

To understand how error correcting codes work we must define the *Hamming Distance* between two binary symbols. Suppose that we have two 6-bit symbols such as 1 1 1 0 1 0 and 1 0 1 1 1 1 then the Hamming distance is the number of digits which are different. So, in this case the Hamming distance is 3. If one bit in a symbol is changed then it has a Hamming distance of one from its original. This change might be due to the action of noise. If two symbols are separated by a large Hamming distance, say 3, then they can still be distinguished even if one bit is changed by noise. The idea of the simplest error correcting codes is to exploit this idea by adding extra digits to a binary symbol, so that the symbols are a large Hamming distance apart.

I will illustrate this with a simple example. The usual symbols for the first eight numbers in binary are:

000 (0) 001 (1) 010 (2) 011 (3) 100 (4) 101 (5) 110 (6) 111 (7)

Many of these symbols are only one Hamming distance apart. For example, a small amount of noise could turn the symbol for a 2 into the symbol for a 3. We now add some extra parity digits to these codes to give the code

000 000 (0) 001 110 (1) 010 011 (2) 011 101 (3)

100 101 (4) 101 011 (5) 110 110 (6) 111 000 (7)

The point of doing this is that each of these codes is a Hamming distance of 3 apart. Suppose that the noise is fairly low and has the effect changing *one bit of a symbol*. If we take the symbol 101 011 for 5, and this changes to (say) 100 011. This is a Hamming distance of one from the original symbol and a distance of at least two from all of the others. So if we receive 100 011 what we do is to find the *nearest* symbol on our list to this. This must be the original symbol, and we *correct* 100 011 to 101 011 to read the symbol without error.

This gives us a simple code which works for eight symbols and a small amount of noise and allows us to correct it by using the following approach

- Receive a symbol.
- Find the closest symbol in our list to the one received.
- Correct the symbol to this one.

All Error Correcting Codes use a similar principle to the above one. A lot of mathematical sophistication is used to find codes for more symbols which will work in the presence of higher levels of noise. Basically, the challenge is to find symbols which are as different from each other as possible. Alongside the development of a code is that of an efficient decoder, which is used to correct corrupted messages fast and reliably.

Examples of such Error Correcting codes are the Reed-Solomon code [4] invented in 1960, which is used in putting music onto CDs, and the earlier Hamming (7,4) [5] code. The Hamming (7,4) code uses three parity digits in a symbol of length 7 binary digits and was invented by Hamming himself in 1950. The first commercial application in mass-produced consumer products appeared in 1982 with the CD where two Reed–Solomon codes are used on each track to give even greater redundancy. This is very

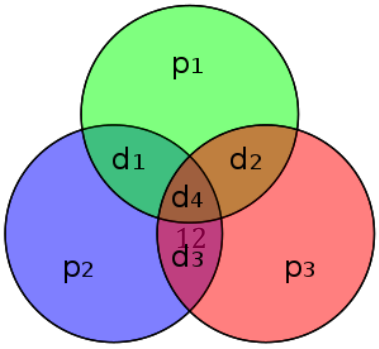
useful when having to reconstruct the music on a scratched CD.

Today, Reed–Solomon codes are widely implemented in digital storage devices and digital communication standards (for example digital TV), although they are now being replaced by low density parity check (LDPC) codes. Reed–Solomon coding is very widely used in mass storage systems to correct the burst errors associated with media defects. This code can correct up to 2-byte errors per 32-byte block. One significant application of Reed–Solomon coding was to encode the digital pictures sent back by the Voyager space probe, which was launched in 1977 and took the first satellite pictures of Jupiter, Saturn and the far planets. A picture of Voyager and an image of Saturn is given below. A major recent user of error correction is Facebook. This is possibly the largest repository of information in the world. It is estimated that 300 million photos are stored on Facebook every day. This information is stored in vast data banks around the world, mostly on spinning discs. Whilst the individual failure rate of a disc is very low, there are so many discs required to store the information that the chance of one of the discs failing at any one time is high. When this happens the data on the disc is recovered efficiently and quickly by using a Reed-Solomon code, so that Facebook can continue without interruption.

Mathsy Bit

A bit of maths here. It is far from simple to work out codes which are both efficient and robust, and a whole branch of mathematics, *Coding Theory*, has been developed to do it.

In the Hamming (7,4) code, 3 parity bits are added to a message with 4 information bits. If x is the vector x = (d1, d2,d3,d4) of the information (which is 4 bits long), then the transmitted symbol y is 7 bits long and is given by y=(p1,p2,d1,p3,d2,d3,d4) where p1,p2,p3 are the parity bits. These parity bits are added so that groups of the digits in the code have an even number of 1s. The combinations of data and parity bits are summarised by the following figure taken from [5].



Suppose that x = (1 1 0 1) then using this figure we see that (p1,p2,p3) = (1 0 0) and hence y=(1 0 1 0 1 0 1). Now suppose that y is incorrectly transmitted so that one bit is corrupted, and we receive instead the signal z= (1 0 1 0 1 1 1). Let's see if we can correct this error. The digits in the green circle have the right parity, and in the blue and red circles they have the wrong parity. The offending digit must therefore be d3 (which is correct).

In a more abstract setting

$$y = G\,x$$

where G is a 7*4 *code generator matrix* and all operations are calculated in modular 2 arithmetic. This is a linear transformation, and the Hamming (7,4) code is an example of a *linear code*. If a corrupted form z of the message y is received, then the bit for which the error is made is found by using a second parity checking 3*7 matrix H so that if d is the vector giving the binary location of the corrupted bit then

$$d = H\,z.$$

Having found d we know where the error has occurred. Then it is easily possible to construct a 4*7 matrix R to recover the original symbol so that

$$x = R\,z.$$

The Reed Solomon code is more sophisticated in its construction. In the classic implementation of the Reed Solomon code the original message is mapped to a polynomial with the terms of the message being the coefficients of the polynomial. The transmitted message is then given by evaluating the polynomial at a set of points. As in the Hamming (7,4) code, the Reed Solomon code is a *linear transformation of the original message*. Decoding works by finding the best fitting message polynomial. All the multiplications for the polynomial are performed over mathematical structures called finite fields. The theory behind the construction of these codes uses advanced ideas from the branch of mathematics called Galois theory, which was invented by the French mathematician Evariste Galois (see below), when he was only 19, and at least 150 years before it was used in CD players.

Data Compression

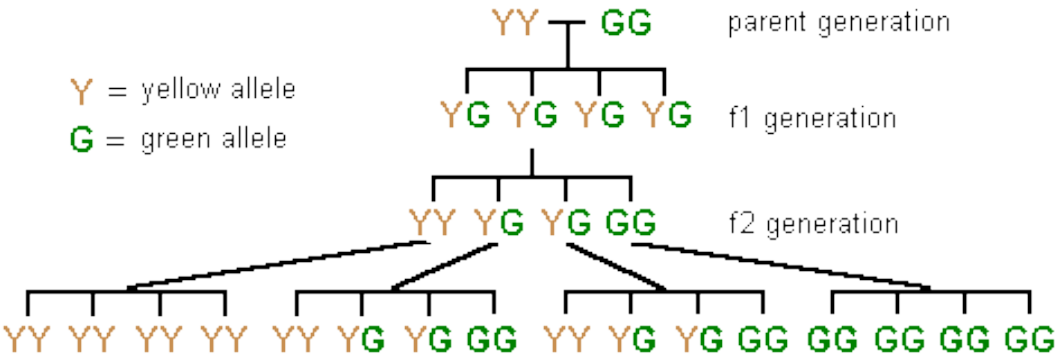
We live in a culture when instead of talking to each other we send a text instead. It is very common when doing this to miss out parts of the message so that it is quicker to send, but still conveys the same information. '*Cn y rd ths mssg whch dsnt hv ny vwls bt hs lts f cnsnts*?' This message has been compressed by missing out the vowels. It can be read easily, but takes up less space than the original. This is a simple example of lossy *data compression* in which less data is transmitted, but the sense of the message is still clear. Without data compression it would take much longer to transmit information, and much harder to store it. We are all very familiar with the products of data compression algorithms. **JPEG** (seen as a .jpg addition to a file name, and short for 'Joint Photographic Experts Group') is a commonly used method of lossy data compression for images particularly for those images taken on a digital camera (say on a mobile phone) and then sent as attachments in emails. JPEG typically achieves 10:1 compression with little loss in image quality. It does this by discarding high frequency features of the image which the human visual systems would not be able to process. This is rather like discarding high notes in music beyond our hearing range. The key to doing this is was originally the Discrete Cosine Transform. This is an amazing mathematical algorithm, which lies at the heart of the modern digital revolution and which was invented by Cooley and Tukey in a seminal paper published in 1965 [6]. The more modern JPEG 2000 standard uses the (wonderful) Wavelet Transform [7]. A similar process is used in the mp3 codes used for audio files.

Genetics, Nature's Digital Code.

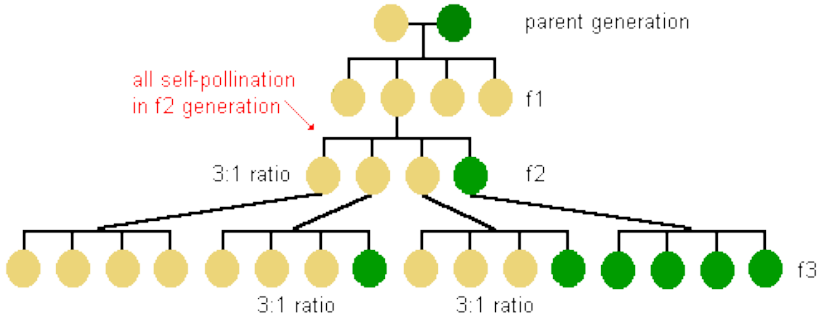
Whilst the amount of information in, say a textbook, might seem formidable, it pales into insignificance when compared with the amount of information needed to produce a living organism. And not only to do this once, but to do it reliably from one generation to the next. To investigate this, we now turn from the technology of human based communication to see how nature passes on the information concerned with inheritance and the resulting evolution of the species. Of course, the most famous figure in this field is Charles Darwin, who published his ground-breaking book 'On the origin of species' in 1859. Despite giving an explanation of how species would change and evolve, Darwin was unaware of the mechanism behind how this actually took place. It was generally thought at the time that evolution occurred by a blending of the characteristics of species, rather like an analogue signal. In contrast, the actual process of inheritance is much more similar to that of *digital transmission* as described above. The nature of this process was discovered in 1866 by the monk Georg Mendel (pictured below). However, the precise mechanism was only identified 100 years later with the discovery of the structure of DNA.

Mendel was looking at how phenotypes (or traits) of a species were inherited from one generation to another. An example of such in humans might be eye colour, or height. Mendel instead looked at the colour of peas. He bred successive generations of peas of different colours and made very careful quantitative measurements of what he observed. In particular he observed that if he took as a parent generation, yellow and green peas and crossed them then all of the next generation were yellow. If he then took this second generation and used this for self-pollination, then he obtained both yellow and green peas in a ratio of three to one (3:1). This already showed a high degree of mathematical regularity. If he then self-pollinated future generations then 1/4 led to yellow peas only, 1/4 to green peas only and ½ to yellow and green, again in the 3:1 ratio. This was evidence of the discrete nature of inheritance rather than a blending of the yellow and green colours. He explained this by

postulating that two factors were responsible for the colour of the pea, a yellow Y and a green G allele. When both were present the Y was dominant over the G and the colour would be yellow. A green pea would only arise if there were two G alleles. The distribution of the alleles was as follows.

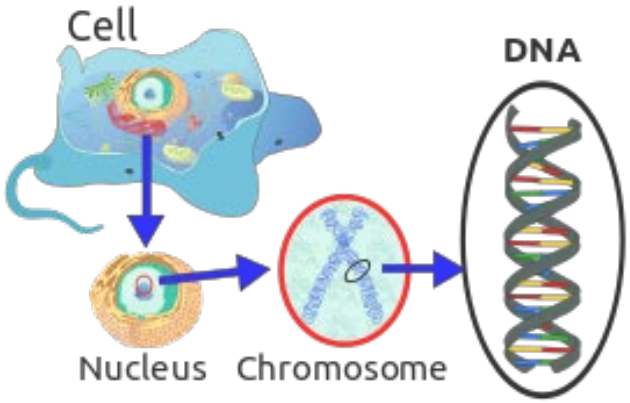


The resulting pea colours with cross pollination followed by self-pollination are then shown below.

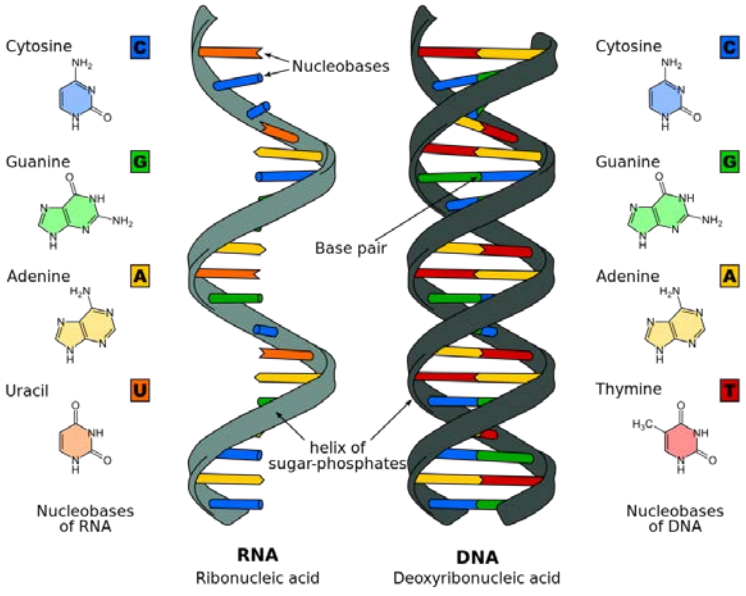


We now know that the mechanism behind this process of inheritance lies in the discovery that discrete amounts of information are coded in your genes, and that traits of an organism, such as the colour of the peas, depended on its sequence of genes. The term *gene* was introduced by Wilhelm Johannsen in 1905. *Deoxyribonucleic acid* (DNA) was shown to be the molecular repository of genetic information by experiments in the 1940s to 1950s. The structure of DNA was studied using X-ray crystallography by Rosalind Franklin, and Maurice Wilkins. This research led James D. Watson and Francis Crick to publish a model of the double stranded DNA molecule as illustrated below. We can already see a strong mathematical regularity in the shape of this molecule, which begs the question of whether this molecule came into existence at random or through the action of deep mathematical principles.

A gene itself is a sequence of the DNA or RNA molecules, which gives the code for a molecule such as a protein that has a function. A chromosome is a long strand of DNA containing many genes. A human chromosome can have up to 500 million base pairs of DNA with thousands of genes. The DNA is first copied into RNA. The RNA can either have a direct function, or, more typically it will be the template for a protein that then goes on to perform a function. The sequence of genes is called the genotype of the individual, and the transmission of genes to that individuals offspring is the basis of the inheritance of its phenotypical traits. Because genes are discrete, we can observe a discontinuous inheritance of the traits of an organism. Some genetic traits are easily visible, such as eye colour or height, and some are not, such as blood type, risk for specific diseases, or internal biochemical processes. The total complement of genes in an organism or cell is known as its genome, which may be stored on one or more chromosomes.



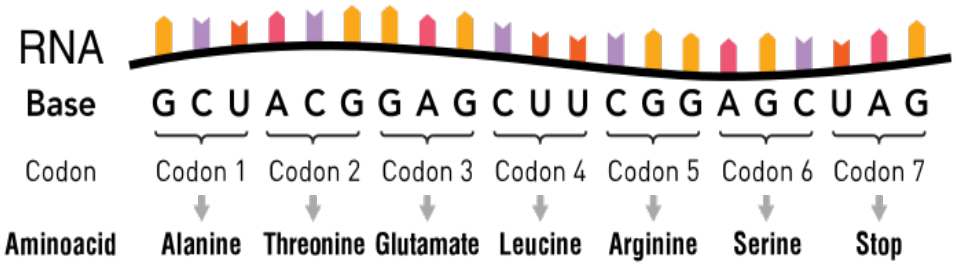
The DNA molecule is a chain made from four types of nucleotide subunits, each composed of: a five-carbon sugar, a phosphate group, and one of the four bases: A adenine, C cytosine, G guanine, and T thymine. The base U uracil is also present in RNA. The components (base pairs) of the genes act rather like the 0s and 1s of a binary message. As we saw, these binary digits can encode a huge amount of information. Similarly, all the proteins in your body are made from protein building blocks called amino acids. There are twenty different amino acids used to make proteins all of which are specified from the 4-letter code sequence above.



Inheritance as a Data Transmission Problem.

Organisms inherit their genes from their parents. Asexual organisms, such as amoebae, inherit a complete copy of the genome of their parents. Sexual organisms (such as us) have *two copies* of each chromosome as they inherit one from each parent and how these combine leads to the traits that we see inherited from each. Genes can acquire mutations in their sequence, leading to different variants, known as *alleles* in the population. These alleles encode slightly different versions of a protein, which also in turn cause different traits. It was Mendel that recognised that some alleles can be dominant, and some can be recessive, which in part is a reason for the large variety that we see in inheritance.

This process of passing on information is done digitally. The RNA code is designed to be read as triplets, and each symbol in the code, called a *codon*. Just about every living thing uses this exact code to make proteins from DNA. These codons are similar to the ASCII symbols, or the other codes we looked at earlier. The codon is *three letters long* taken from the A C G U base pairs for RNA. These are 4*4*4 = 64 possible codons and these are then in turn used as the codes for 20 different amino acids. There are also special "start" and "stop" codons that mark the beginning and end of a gene. A single gene makes a protein as follows. Genes have segments called exons which can join together in various different combinations when they are then transmitted into the (messenger) RNA. These segments of RNA guide the assembly of amino acids in long sequences. The amino acid chains then fold into proteins that go on to perform different functions. These sequences in each codon for a typical gene are illustrated below. With 64 different possibilities the code has a degree of redundancy. That is, most of the 20 amino acids have at least two different codons, giving a measure of error correction.



Again, we see a high degree of mathematical regularity here. The codons are playing a similar role in the process of inheritance to the symbols in the codes that we looked at earlier in this lecture. The use of this process of transmitting information is very effective leading to errors in the order of one in 100 million. Does nature use other forms of error correction? At some level it must do, otherwise species could not reproduce reliably. However, fortunately (for us) this error correction allows for mutations to occur. Otherwise evolution would not be possible, and we would not be here. However, this is a subject for a future lecture. For much more information about genetics see [8].

In conclusion, we live in an information age, and the mathematical development of digital data transmission, combined with error correcting codes, allows us to communicate accurately and reliably. The processes used by nature to transmit information from one generation to another are themselves highly reliable and in many ways, resemble digital communication methods. I hope that you will agree with me that *Maths is truly coded in our genes.*

References

[1] C.E. Shannon, *A mathematical theory of communication,* Bell System Tech J., 27, (1948), 379-423.

[2] https://en.wikipedia.org/wiki/Information_theory

[3] J. Leibnitz, *Explication de l'Arithmétique Binaire*, Academie royale des sciences, (1703).

[4] I. Reed and G. Solomon, *Polynomial Codes over Certain Finite Fields*, J. Soc. Indust. Appl. Math, 8, (1960), 300-304.

[5] R. Hamming, *Error detecting and error correcting codes*, Bell System Tech. J., 26, (1950), 147-160.

[6] J. *Cooley and J. Tukey, An algorithm for the machine calculation of complex Fourier series, Math Comput, 19, (1965), 297–301.*

[7] C. Burrus, H Guo, and R. Gopinath, *Introduction to Wavelets and Wavelet Transforms: A Primer*, (1998), Prentice Hall.

[8] S. Jones, *Genetics for beginners*, (1993), Icon Books