shannon's theorem
... and one of the homework problems

■ 7.8 - an example code with information rate > 0.6

Suppose that the sender wishes to code blocks of size k with words of length n, using a 1-error correcting code. If it is required to transmit information with an information rate of not less than 0.6, what are the smallest possible values of n and k ?

The parameters are

n	number of bits in a codeword
k	number of bits of information in that codeword

information rate $= \rho \equiv \frac{k}{n} \geq 0.6$

r	error correct = 1
δ	min hamming distance between codewords = $2r + 1 = 3$

For $r = 1$, we need a each codeword to exist in its own neighborhood of $(n+1)$ words, so we have the packing result

$$|C| \, (1+n) \leq 2^n$$

Since we need a codeword for each plaintext block of k bits,

$$|C| = 2^k$$

Therefore

$$2^k \, (1+n) \leq 2^n$$

or

$$1 + n \leq 2^{n-k} = 2^{n\left(1-\frac{k}{n}\right)} \leq 2^{n(1-\rho)}$$

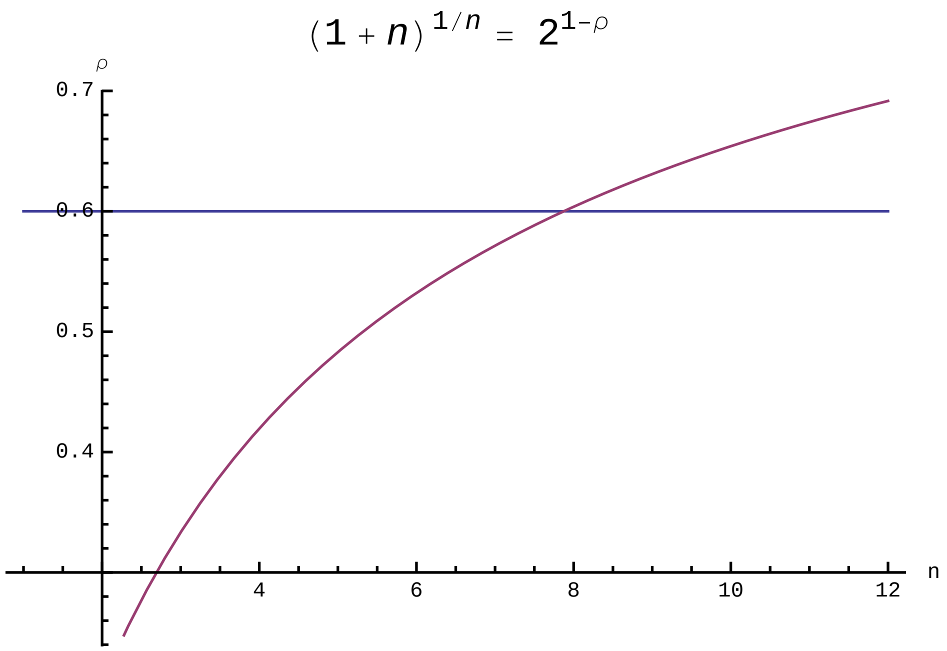
or

$$(1+n)^{1/n} \leq 2^{1-\rho}$$

Note that for any given value of ρ, we can always choose a value of n large enough to make this inequality true.

The picture looks like this.

```
Plot[{0.6, 1 - Log[2, (1 + n)^(1/n)]}, {n, 1, 12},
 AxesLabel → {"n", "ρ"}, PlotLabel → "(1 + n)^(1/n) = 2^(1-ρ)"]
```



The exact value of *n* where these two curves cross is

```
FindRoot[((1 + n)^(1/n) - 2^(1-ρ)) /. ρ → 0.6, {n, 8}]

{n → 7.87393}
```

But of course we need *n* to be an integer, so it must be at least 8.

First we try $n = 8$ ... which as it turns out won't work. In that case, *k* is at least $\rho\, n = 0.6 \times 8 = 4.8$, which means $k = 5$ or bigger.
Then the packing inequality is violated :

```
2^k (1 + n) /. {k → 5, n → 8}

288

2^n /. {n → 8}

256
```

$(2^k (1+n) \leq 2^n) /. \{k \to 5, n \to 8\}$

```
False
```

Another way to see that these values won't work is to look at the Hamming codes, which give the "best" 1-error correcting codes. With $(n, k) = (8, 5)$, we have $n - k = m = 3$. The hamming code with $m = 3$ has only

$2^m - 1 = 7$ independent basis vectors, not 8. Or if we try to keep $m = 3$ with $(n, k) = (7, 4)$, then the information rate of $4/7 = 0.57$ is too low.

The next value to try, then, is $n = 9$. Then k is at least $\rho\, n = 0.6 \times 9 = 5.4$, so $k = 6$. Again we check the packing inequality :

```
(2^k (1 + n) ≤ 2^n) /. {k → 6, n → 8}
```

```
False
```

And again we strike out. Time to get systematic.

```
Do[
 ρ = 0.6;
 k = Ceiling[n ρ];
 Print["{n,k}=", {n, k}, " gives 2^k(1+n)≤2^n is ", (2^k (1 + n) ≤ 2^n)],
 {n, 4, 12}
]
```

```
{n,k}={4, 3} gives 2^k(1+n)≤2^n is False

{n,k}={5, 3} gives 2^k(1+n)≤2^n is False

{n,k}={6, 4} gives 2^k(1+n)≤2^n is False

{n,k}={7, 5} gives 2^k(1+n)≤2^n is False

{n,k}={8, 5} gives 2^k(1+n)≤2^n is False

{n,k}={9, 6} gives 2^k(1+n)≤2^n is False

{n,k}={10, 6} gives 2^k(1+n)≤2^n is True

{n,k}={11, 7} gives 2^k(1+n)≤2^n is True

{n,k}={12, 8} gives 2^k(1+n)≤2^n is True
```

So the smallest ones that work are $\{n, k\} = \{10, 6\}$.

The problem doesn't ask for it, but let's construct such a code, just for clarity.

The generating matrix will therefore have 10 rows and 6 columns. In standard form, the top 6 by 6 is the identity matrix, and the bottom 4 by 6 must be made of columns with at least two 1's, so that each column has weight 3. Constructing one of these isn't hard. For example, here's one with the desired properties :

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

The corresponding check matrix, also in standard form, would be

$$H = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Is this a hamming code? No; then H would be 4 rows by $2^4 - 1 = 15$ columns.

Is it a perfect code? No; there are some words which are more than 1 away from any codeword.

- **Excersise: find a codeword that's more that 1 away.**

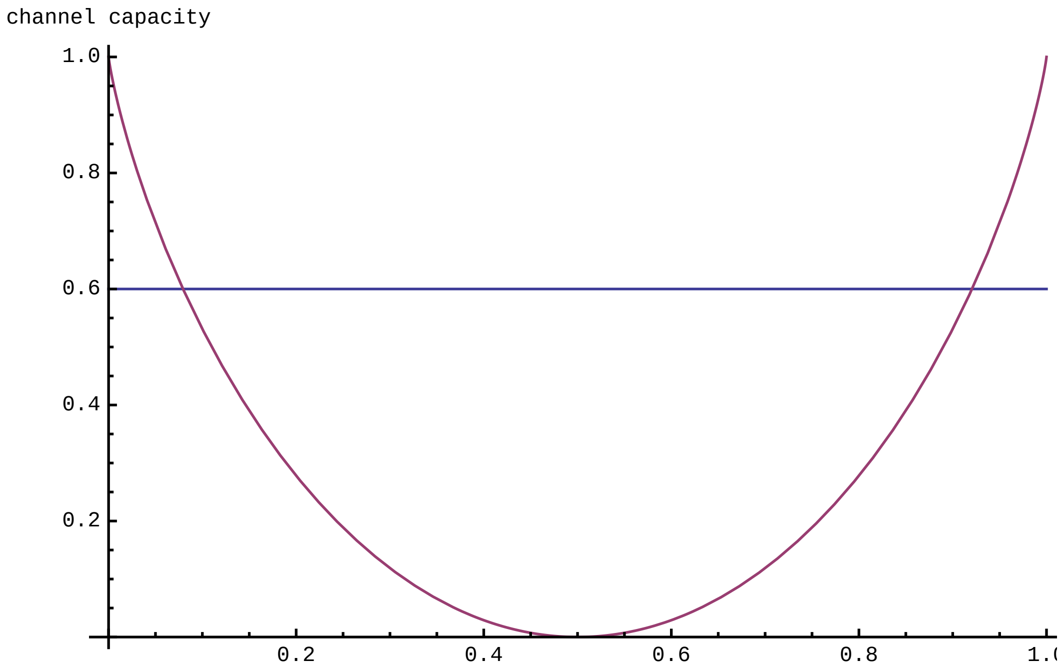
Shannon's Theorem

7.8 - how big an error rate is under Shannon's value ?

So, let's get back to a concrete example.

The code introduced from exercise 7.8 has a known information rate. What is the corresponding bit error probability from Shannon's Theorem? Presumbably this code (and other ρ = 0.6 with bigger n values) can do something reasonable with this error rate.

```
Plot[{0.6, capacity[e]}, {e, 0, 1},
 AxesLabel → {"bit error prob", "channel capacity"}]
```



```
eValue = FindRoot[capacity[e] == 0.6, {e, 0.1}]
```

```
{e → 0.0793826}
```

For this code, $n = 10$, so each codeword sent has 10 bits. What are the probabilities of getting a given number of errors? Well, we get a binomial expansion :

prob(no errors) = $(1-\epsilon)^{10}$
prob(1 error) = $10\,\epsilon^1\,(1-\epsilon)^9$
prob(2 errors) = $(10 \times 9/2)\,\epsilon^2 (1-\epsilon)^8$
... and so on.

The whole thing looks like this :

```
Table[{m, Binomial[10, m] e^m (1 - e)^(10-m)} /. eValue, {m, 0, 10}] //
 MatrixForm
```

$$\begin{pmatrix} 0 & 0.437312 \\ 1 & 0.377084 \\ 2 & 0.146318 \\ 3 & 0.0336443 \\ 4 & 0.00507687 \\ 5 & 0.000525319 \\ 6 & 0.0000377475 \\ 7 & 1.85993\times 10^{-6} \\ 8 & 6.01414\times 10^{-8} \\ 9 & 1.15241\times 10^{-9} \\ 10 & 9.93695\times 10^{-12} \end{pmatrix}$$

Since we can correct 0 or 1 bit errors, the probability of a mistake is (1 - prob(0) - prob(1)) :

```
approxMistakePerWordProbability = 1 - ((1 - e)^10 + 10 e (1 - e)^9) /. eValue
```

```
0.185604
```

I should note that this analysis isn't quite right.

In particular, the model for the source, which was equal probabilities of 0's and 1's, isn't quite what we're sending: we're sending codewords, which are made up of specific patterns of 0's and 1's, so our capacity calculation is off.

To do a more accurate job, we'd need to have a model of the probability of codewords (say, the same probability for each), and then from that and the encoding matrix calculate the probability of 1's and 0's, and then from that calculate the channel capacity. Any movement of the probabilities p(0) and p(1) away from 50/50 will give us a lower channel capacity for a given symmetric bit error rate.

■ What happens when the information rate is above the channel capacity ?

Here's what this one looks like for some other mistake rates .
(All these have the same information rate = 0.6 and n=10.)

```
mistakeProb[e_] := 1 - ((1 - e)^10 + 10 e (1 - e)^9);
```

```
Table[{e, capacity[e], mistakeProb[e]}, {e, 0.05, 0.3, 0.05}] //
 MatrixForm
```

```
⎛0.05 0.713603 0.0861384⎞
⎜ 0.1 0.531004  0.263901⎟
⎜0.15  0.39016    0.4557⎟
⎜ 0.2 0.278072   0.62419⎟
⎜0.25 0.188722  0.755975⎟
⎝ 0.3 0.118709  0.850692⎠
```

With these codewords, the probability of a mistake starts to grow pretty quickly as the probability of a bit error increases.