THE BIRTHDAY PARADOX AND RANDOM NUMBER GENERATION

Derek O'Connor*

June 18, 2011†

1 The Birthday Paradox

The Birthday Paradox or problem asks for the probability that in a room of n people, 2 or more have the same birthday (not date), assuming all years have $N = 365$ days. It is called a paradox because most people are surprised by the answer when there are (say) 30 people in the room.[1]

We treat the birthdays as a sample of size n from a population of size N, *with replacement*. There are $N = 365$ possible values for each person's birthday, hence there are $N \times N \times \cdots \times N = N^n = 365^n$ possible ordered sets $(b_1, b_2, \ldots, b_n)$ of n birthdays. We assume that all of these are equally likely, with probability $1/N^n = 1/365^n$.

It is easier to calculate the probability of the complement, so let us calculate the probability that no two birthdays are the same, i.e, that the ordered set $(b_1, b_2, \ldots, b_n)$ has no duplicates. This restriction means that b_1 can have $N = 365$ different values, b_2 can have $N-1 = 364$ different values, …, b_n can have $N-n = 365-n+1$ different values. Hence the number of ordered sets on n birthdays under the no duplicate restrictions is $N \times (N-1) \times \cdots \times (N-n+1) = 365 \cdot 364 \cdot 363 \cdots (365-n+1)$. The probability for each ordered set is $1/N^n = 1/365^n$ and so the probability of the total is

$$\begin{aligned} \Pr(\text{No birthdays the same}) &= \frac{N \times (N-1) \times \cdots \times (N-n+1)}{N^n} \\ &= \frac{365 \cdot 364 \cdot 363 \cdots (365-n+1)}{365^n} \end{aligned} \tag{1.1}$$

Let us denote the probability in (1.1) as $P(N,n)$, then the probability that there are two or more people with the same birthday is $1 - P(N,n)$.

*Web: http://www.derekroconnor.net Email: derekroconnor@eircom.net

†Started: 12 Dec 2010.

[1] Years ago it was easy to test this answer in a pub by doing a quick survey. In the mid-1970s I provoked a friend of mine, Barry Walsh, to do a survey in Nick's English Hut, a student bar in Bloomington, Indiana. He came back after 5 minutes, shaking his head: he had found 4 people with the same birthday. It's a bit pointless in my local these days – you're lucky to get 4 people in on a week-night. I could make it 5, I suppose.

As it stands, the formula (1.1) is difficult to compute. But before we look at simplifications of it, let us look at a few values that have been tabulated in Mosteller, *et al*.[2] Looking at

Table 1. THE BIRTHDAY PARADOX

n = Number in room	5	10	20	23	30	40	60
Probability that two or more Birthdays are the same	0.027	0.117	0.411	0.507	0.706	0.891	0.994

the Table 1 we see that there is a 50:50 chance of two birthdays the same in a room of 23 people. Most people would give much lower odds: 100:1, 200:1, or even 365:1. Hence the 'paradox'. [3]

We wish to simplify the formula for $P(N,n)$ so that it is more amenable to calculation and symbolic manipulation. Following Feller[4], we have

$$P(N,n) \;=\; \frac{N(N-1)\cdots(N-n+1)}{N^n} \;=\; \left(1-\frac{1}{N}\right)\left(1-\frac{2}{N}\right)\cdots\left(1-\frac{n-1}{N}\right). \quad (1.2)$$

If n is small relative to N, then all cross products can be neglected and we have

$$P(N,n) \;\approx\; 1-\frac{(1+2+\cdots+n-1)}{N} \;=\; 1-\frac{n(n-1)}{2N}, \quad \text{for small} \quad n. \quad (1.3)$$

As n increases, $P(N,n)$ decreases to 0, and Feller uses (1.3) and the approximation $\log(1-p)\approx -p$, for small p, to get:

$$\log P(N,n) \;\approx\; -\frac{(1+2+\cdots+n-1)}{N} \;=\; -\frac{n(n-1)}{2N} \approx -\frac{n^2}{2N}, \quad \text{for large} \quad n. \quad (1.4)$$

Formulas (1.3) and (1.4) are much easier to work with and calculate than (1.2) and we can give this approximate probability distribution function for the number of duplicates $N_d(N,n)$ in a random sample of size n from a population of size N:

$$\boxed{\Pr(N_d(N,n)>0) \;=\; 1-P(N,n) \approx 1-e^{-\frac{n^2}{2N}}.} \quad (1.5)$$

From this approximation we get

$$\Pr(N_d(N,n)>0) = \frac{1}{2}, \quad \text{for} \quad n \approx 1.1774\sqrt{N}. \quad (1.6)$$

It must be remembered that equations (1.5) and (1.6) are valid for large values of n and N only.

[2] Mosteller, Rourke, and Thomas, *Probability with Statistical Applications*, Addison-Wesley, 1961.

[3] This example points to a more general problem: most people, even those in the 'professions', are very bad at estimating probabilities and understanding statistical data. Medical researchers are notorious for misusing and abusing statistics. The Sally Clarke case was one of the most egregious misuses of probability and statistics – a nasty combination of Medicine and Law. See http://en.wikipedia.org/wiki/Sally_Clark

[4] William Feller, *An Introduction to Probability Theory and its Applications*, Vol.1, 3rd Ed., Wiley, 1968, p 33.

2 MATLAB's Birthday Problem

Random number generators have many uses, ranging from the approximate calculation of complicated integrals and sums to large-scale cosmological simulations.

Many applications require long sequences (or large vectors) of random numbers. In MATLAB these are supplied by the simple statement `r = rand(n,1)`. This statement fills the vector $r[1:n]$ with random numbers uniformly distributed on $(0,1)$.

We wish to answer the question:

> What is the probability that duplicates occur in `r = rand(n,1)`, as n gets large ?

Should we expect to get duplicates in vectors generated by MATLAB's `r = rand(n,1)`, if we have enough time and memory? After all, a vector of size 10^8 is no longer considered huge, when many people have machines with 8 cores (mainly idle) and 16GB of memory.

MATLAB's Uniform Random Number Generator

The default random number generator in MATLAB Versions 7.4 and later is their implementation of the Mersenne Twister algorithm by Nishimura and Matsumoto.[5] This is what the documentation says:

> The `rand` function now supports a method of random number generation called the Mersenne Twister. The algorithm used by this method, developed by Nishimura and Matsumoto, generates double precision values in the closed interval $[2^{-53}, 1-2^{-53}]$, with a period of $(2^{19937}-1)/2$

There are $2^{53}-1$ binary numbers (bit strings) in the closed interval $[2^{-53}, 1-2^{-53}]$ Hence $\Pr(U=u) = 1/(2^{53}-1) \approx 2^{-53}$. Thus the generator may be thought of as a discrete-valued random variable U whose probability mass function is

$$p_{\text{rand}}(u) = \Pr(U=u) \approx \frac{1}{2^{53}}, \quad u \in \{\text{all 53-bit strings, except } 0\}. \tag{2.1}$$

Notice that the MATLAB's `rand` *never* generates 1, nor any value less than 2^{-53}, including 0. This may come as a surprise to some people.

Now we can calculate the probability that rand(n,1) has one or more duplicates. This probability is $\Pr(N_d > 0)$, which is calculated using equation (1.5) with $N = 2^{53}-1$, and is plotted in Figure 1.

We can see that after $n = 10^7$ this probability starts to increase significantly, and has converged to 1 beyond $n = 10^9$. We have from (1.6) that $\Pr(N_d > 0) = 1/2$ at $n = 1.1774\sqrt{N} \approx 1.12 \times 10^8$. Vectors of this length (and longer) are not uncommon today, and are bound to increase in the future.

[5] http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

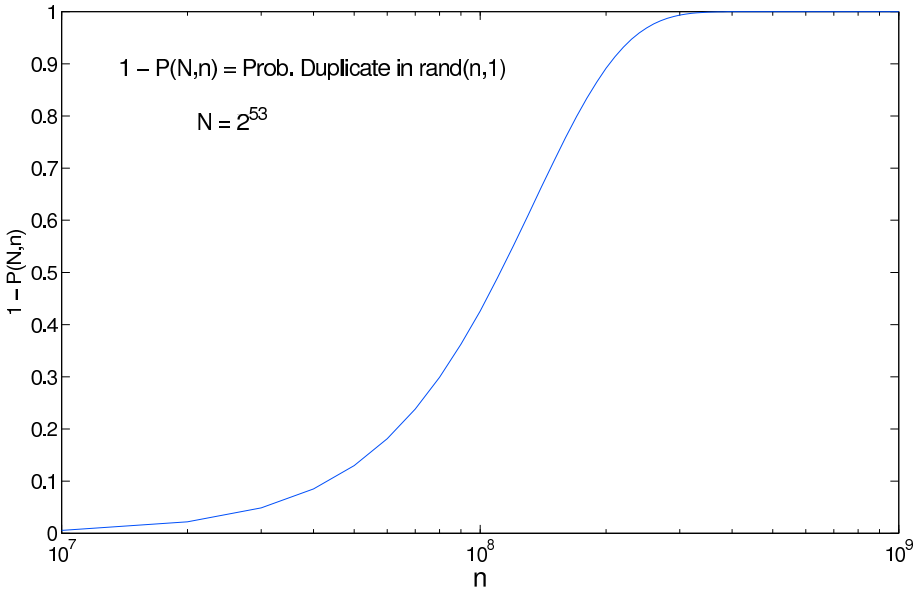

Figure 1. PROBABILITY OF DUPLICATES IN A RANDOM VECTOR OF SIZE n

The expected number of duplicates N_d is approximately[6]

$$\mathbb{E}(N_d) \approx \frac{n^2}{2N}, \quad \text{and} \quad \text{S. Dev}(N_d) \approx \frac{n}{\sqrt{2N}}. \tag{2.2}$$

Hence, we can expect to get one duplicate when $n = \sqrt{2N} = \sqrt{2^{54}} = 1.34217728 \times 10^8$

Testing the Theory. The theory is simply checked by the function `FindDupsRand(n)`. The results are shown in Table 2, which confirms the theory.

```
function [S,ndup,duposn] = FindDupsRand(n);
  Testing rand(n,1) for duplicates
  Derek O'Connor, 14 Dec 2010
  duposn = zeros(200,1);                    The positions of duplicates
  S = rand(n,1);
  S = sort(S);          S is sorted, duplicates will be in contiguous pairs
  ndup = 0;
  for k = 2:n
      if S(k) == S(k-1)
          ndup = ndup+1;
          duposn(ndup) = k-1;
      end;
  end;                          Don't bother checking for triples - unlikely.
  return;
  end; FindDupsRand
```

[6] See Appendix, Math Forum answer.

Table 2. BIRTHDAY TESTS OF `rand(n,1)`, 5 RUNS FOR EACH n.

n	Prob. Dups	No. Duplicates	Avg. Dups.	E(dups)	$T_g + T_s + T_d = T$
$1 \cdot 10^8$	0.4259917	1, 0, 1, 0, 1	0.6	0.56	$2 + 4 + 2 = 9$
$2 \cdot 10^8$	0.8914393	4, 2, 1, 1, 6	2.9	2.22	$4 + 9 + 4 = 17$
$3 \cdot 10^8$	0.9932351	5, 4, 6, 5, 3	5.6	5.00	$6 + 13 + 6 = 25$
$4 \cdot 10^8$	0.9998611	12, 14, 5, 7, 13	8.2	8.88	$8 + 20 + 8 = 36$
$5 \cdot 10^8$	0.9999991	18, 18, 18, 16, 20	16.0	13.9	$10 + 24 + 10 = 44$
$1 \cdot 10^9$	1.0000000	66, 64, 43, 48, 62	56.6	55.5	$42 + 73 + 20 = 135$

T_g, T_s, T_d are the times (secs) to generate, sort, and test the random vector, respectively.

3 Conclusion

Programmers and the Birthday Paradox. We have shown that the probability of getting duplicates in a double precision random vector of length n, is high for $n > 10^8$. This is not a deficiency in MATLAB's Mersenne Twister random number generator or any other generator, but the inevitable result of using any 53-bit generator to generate long vectors (and has nothing to do with the period). Quite simply, it is the Birthday Paradox for random number generators.

The original birthday problem is called a paradox because people are surprised that the value of n is so low for a 50:50 chance of getting two birthdays the same. This prompts the question: *Are programmers surprised that there is a 50:50 chance of getting duplicates in a random vector of size* $n \approx 10^8$*?*

It is undoubtedly true that some programmers assume implicitly that no duplicates occur, and write programs accordingly. Because the assumption is implicit it would be difficult for anyone, including the original programmer, to determine where in the program this assumption comes into play.

The main purpose of this note has been to raise the awareness of such programmers to the possibility of duplicates and hope that they program accordingly.

Tests performed on a DELL PRECISION 690
with
2×Quad-Core INTEL XEON Processors 5345, 2.33GHz, 16GB RAM
using
WINDOWS 7 Professional(64 bit) SP1, and MATLAB R2008a(64bit).

Appendix

Math Forum – Probability of Duplicate Pairs

Date: 05/13/2003 at 07:23:54
From: Neil
Subject: Probability of generating duplicate addresses

I am trying to find the probability of getting at least 20 duplicate addresses when I draw a sample of 30,000 at random from UK households (estimate 21,000,000) where there is replacement every time a selection is made.

I have managed to work out the probablity of getting at least one duplicate by subtracting the probability of getting no duplicates from 1. This is extremely high - coming in very close to 1. However, I cannot see how to calculate $2+$, $3+$ duplicates etc.

I have used the basic formula of $1-[(N-n)/N]*[(N-n-1)/(N-1)]\ldots$ where N is the number in the universe, n is the number in the sample, and have repeated this n times to reflect the number of selections.

Date: 05/13/2003 at 11:07:21
From: Doctor Mitteldorf
Subject: Re: Probability of generating duplicate addresses

Neil -

A quick and practical answer to your question is this: There are roughly $(1/2)30,000^2$ pairs in your sample. If the pairs were independent, the probability for each one to be a match would be $1/21,000,000$. So the mean number of expected matches would be

$$\frac{1}{2}\frac{30,000^2}{21,000,000}=21 \tag{3.1}$$

The standard error in this number is roughly its square root, so you'd expect 21 ± 5 duplicates in your 30,000 draws. Because the numbers are so high, this is quite an accurate estimate.

You might verify the estimate by a numerical experiment: write a program to draw 30,000 random numbers between 1 and 21,000,000, then sort the numbers and count the duplicates. Repeat 100 times, and you'll have a good idea whether the above estimate is valid.

Here are some thoughts on the formal (exact) solution to your problem, which I formulate thus: Suppose you draw n samples from a universe of N objects with replacement. What is the probability of exactly r duplicate pairs?

It is easier to approach this problem by counting permutations separately, even though in the final analysis order doesn't matter. The number of possible samples of n is then just N^n. The subset in which these are all different numbers $N!/(N-n)!$, so the probability

for $r = 0$ duplicates is

$$\Pr(r = 0 \text{ duplicates}) = \frac{N!}{(N-n)!N^n} \tag{3.2}$$

Suppose there are r duplicate pairs. Then the remaining $(n-2r)$ samples can be selected (and ordered) from the remaining $(N-r)$ universe objects in

$$\frac{(N-r)!}{((N-r)-(n-2r))!} = \frac{(N-r)!}{(N-n+r)!}$$

different ways.

Where are the duplicate pairs located in the sample of n? This is the number of ways of choosing $(2r)$ from a list of $n : C(n,2r)$.

There are $C(N,r)$ ways of choosing which r objects will be the duplicates.

Finally, how many different ways may the r duplicates be arranged to make a list of length $2r$? The answer is $(2r)!/2^r$. This is because there would be $(2r)!$ permutations of the list if all the members were distinct, but each swapping of a pair means we've over-counted by a factor of 2.

Putting all this together, the probability of exactly r distinct pairs (with no triples or quadruples) is

$$\frac{(N-r)!n!N!(2r)!}{(N-n+r)!(n-2r)!(2r)!(N-r)!r!2^rN^n} = \frac{N!n!}{(N-n+r)!(n-2r)!r!2^rN^n} \tag{3.3}$$

Practicing what I preach, I've tested this formula in 1,000,000 numerical trials for $N = 30$ and $n = 15$, $r = 0,1,2,3,4$, and find that it correctly predicts the number of pairs.

- Doctor Mitteldorf, The Math Forum
http://mathforum.org/dr.math/