Chapter

X

The Birthday Paradox

The *Birthday Paradox* [1] is a well-known example of the non-intuitive nature of probability and statistics. The 'paradox' is that although there are 365 days in a year (not counting leap years), when you have a group of 23 people, there is slightly more than an even chance (50.7%) that two or more of those people will share a common birthday (month and day). While there is nothing 'paradoxical' about this phenomenon, it is certainly surprising as there are 365 days in the year, and one might expect a much lower likehood of shared birthdates in such a small crowd. As the number of people increases, the likelihood of shared birthdays also grows - and rapidly. Figure X.1 shows how the likelihood of a match increases with the number of people in the group. For 50 people, there is a 97% chance of a match, while for 89 people, there is a 99.999+ % chance of a shared birthday – pretty good odds!

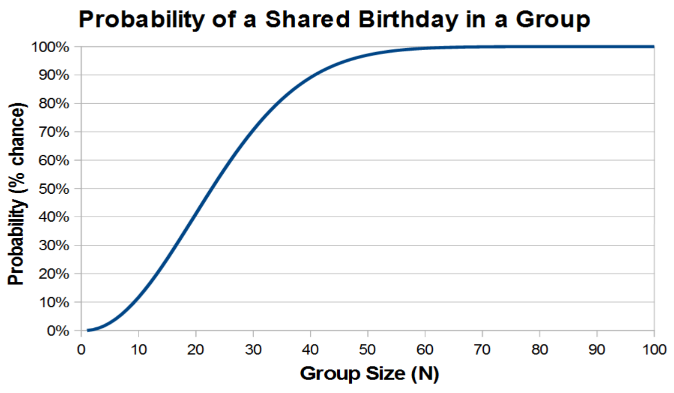

Figure X.1

An explanation for this apparent 'paradox' can be found through a simple application of probability theory. One of the most straighforward approaches to calculating the probability of coincidence is to first calculate the opposite - the probability that people *don't* share a common birthday. For the case of two people, the likelihood that they do not share a birthday is 364/365, as the first person's birthday falls on some given day (we don't care which), and the second person's birthday therefore has 364 opportunities to not coincide with the first person's. If we consider a third person (N=3), he or she has only 363 opportunities not to share a date with the first two – who we already are assuming have birthdays that fall on different days. By induction, we can generalize this into an expression (eqn. X.1) for the probability (which ranges from 0 to 1) that in any group of N people, *none* share birthdays:

$$P_{Not\ Share} = \frac{365-1}{365} \times \ldots \times \frac{365-(N-1)}{365} \qquad \textbf{eqn X.1}$$

Once we know the probability that no two people in the group share a birthday, obtaining the probability that at least two people *do* share a birthday can be found by simply subtracting the above expression (eqn X.1) from '1', yielding equation X.2:

$$P_{Share} = 1 - P_{Not\ Share} = 1 - \frac{365-1}{365} \times \ldots \times \frac{365-(N-1)}{365}$$

(eqn. X.2)

Of course, there are some assumptions behind the above analysis. The first is that we are completely ignoring people born on February 29th, which only occurs on leap years (every 4 years). The second, and more significant assumption is that birthdays are uniformly distributed among the 365 days of the year, which is definitely *not* the case in at least some places [2]. An uneven distribution, however, can work in favor of more shared birthdays.

While deriving the the above formula for describing the probability of two people in a group sharing a birthday is relatively straightforward, deriving an expression for the probability of three or more people having a common birthday is considerably more difficult – but it is possible and has been done [3]. The expressions, however, are quite complex and involve calculating large factorials, which can be numerically challenging. Using a simulation model to obtain a probability estimate, however, is a much simpler approach to solving this problem – although the downside of simulation is that we will lose the exactitude of an analytic result and have to make do with approximate solutions.

So how does one go about building a simulation model for estimating probability? Consider the case of *measuring* the probability that a coin toss will come up heads, as opposed to just assuming a value (say 50%). A single coin toss gives you a heads/tails result – not a probability value. To estimate the probability of getting heads requires two things:

1) An 'experiment' you can perfom that yields heads or tails (not heads). This is a single coin toss.
2) A process that performs a statistically meaningful number of iterations of the above experiment, from which you can estimate the probability of getting heads – the number of times heads appears out of the total number of experiments (tosses).

In the case of our birthday problem, the 'experiment' that needs to be done is to consider a group of 'N' people with randomly distributed birthdays, and then check whether 'M' of them share a common birthday. This experiment will yield a result of 'true' if M people share a birthday, and 'false' if they do not. For example, if N=35 and M=3, the experiment will look at the birthdays of 35 people, and only return a 'true' result if three (or more) share a date.

While it is certainly possible to perform this 'experiment' in the real world by roudning up a group of people and querying them all for their birthdays, the goal here is to build a simulation model of the experiment that is (presumably) a lot less effort to run. One way of implementing this simulation experiment is shown in Figure X.2, the function ***RunExperiment***:

```
RunExperiment(N as integer, M as integer) as boolean

      // Inputs:   N - number of people in group
      //           M - Number of people with same birthday
      // Returns:  TRUE if M or more people share a
      birthday,
      //           FALSE if fewer than M people share a
      birthday

      dim R as new random

      dim birthdaycount(365) as integer

      dim i as integer, day as integer

      for i = 1 to N

        day = R.InRange(1,365)   ' Random day from 1..365
        birthdaycount(day) = birthdaycount(day) + 1
        if birthdaycount(day) >=M then return TRUE

      next i

      return FALSE  ' Matches not found
'END
```

Figure X.2 – Code for Function RunExperiment

Function ***RunExperiment*** first defines, and celars an array of 365 counters, corresponding to days of the year (ignoring the leap year question completely). It then models the group of ***N*** people by sequentially selecting N random year-days in the range of 1...365 using the ***InRange*** function of random number generator object ***R***. For each of those random dates, ***RunExperiment*** increments the corresponding day counter, and exits with a return value of ***true*** if any of the counts equals or exceeds **M**. Note that this can occur before all ***N*** birthdates have been tallied if there is a sufficient matched set. Finally, if no suitable match sets have been found, the function returns a value of ***false***.

Now that we have the algorithmic equivalent of a coin-toss experiment for our birthday problem, the next step is to iterate the experiment, collect a tally of true results versus the total number of trials, and report the results. Since the expected results from our experiment are just a list of numbers, this doesn't require a graphic user interface (GUI) and can be implemented as the command-line program shown in Figure X.3

```
Application.Run(args() as string) as Integer
      const TRIALS = 10000
      const N_MAX = 100
      const M = 2

      dim N as integer, trial as integer
      dim truecount as integer

      print "N    P"
      for N = M to N_MAX
        truecount = 0
        for trial = 1 to TRIALS
          if RunExperiment(N, M) then truecount =
                   truecount+1
        next trial
        print format(n,"##0") + "  " +
      format(truecount/TRIALS, "0.000000")
      next N

      call input   ' wait for user input -
                   ' keeps text window from
                   ' closing at end of program
'END
```

Figure X.3– Top level of Birthday Simulation Program

There are several things going on in the ***Application.Run*** function. First, ift contains an outer loop that iterates values of N from M to N_MAX – enabling the function to print a table of probabilities for a range of N values. Next, in its inner loop, executes the ***RunExperiment*** function ***TRIALS*** times (10000), keeping track of the number of times there is a set of ***M*** birthday matches. If you run the program, you get an output like the following (Figure X.4, truncated for brevity):

```
N   P
2  0.0018000
3  0.0084000
4  0.0170000
5  0.0250000
6  0.0395000
7  0.0562000
8  0.0759000
9  0.0963000
10  0.1179000
11  0.1454000
12  0.1725000
13  0.1903000
14  0.2212000
      .
```

Figure X.4 – Sample Output of Program, M=2

Now that we have a simulation model that executes and provides output, the next questions are (1) whether the model it embodies reflect reality to a useful degree and (2) whether it is implemented correctly. For this system, we are in luck in the verficiation and validation departments, as we have already developed an analytic solution for the case of M=2 (two people share birthdays). For this case, we can the simulation model by running it and comparing the simulation results to the known analytic results at some number of points, as shown in Table X.1.

N	P Simulated	P Analytic	Difference
2	0.0030	0.0027	0.0003
3	0.0091	0.0082	0.0009
4	0.0164	0.0164	0.0000
5	0.0304	0.0271	0.0033
20	0.4130	0.4114	0.0016
21	0.4334	0.4437	-0.0103
22	0.4794	0.4757	0.0037
23	0.5042	0.5073	-0.0031
24	0.5475	0.5383	0.0092
25	0.5651	0.5687	-0.0036
35	0.8192	0.8144	0.0048
36	0.8314	0.8322	-0.0008
37	0.8437	0.8487	-0.0050
38	0.8644	0.8641	0.0003
39	0.8800	0.8782	0.0018

Table X.1 – Simulated vs Analytic Results for Selected N , M=2

You can see that the simulated results are close to the analytic ones – close enough to provide at least tentative credibility. Assuming the model is good, the questions now are whether the model is accurate enough to be useful, and if not, how can we improve it?

The notion of of whether a simulation model is 'accurate enough' is driven by the purpose for which it is employed. In the case of the birthday problem, the purpose of the model is ultimately to have something to amaze people with at a gathering – if the probability estimates are off by a few percent, the worst-case end results may be a little embarassment at being wrong. In contrast, a model used to provide probability estimates for use in a high-frequency stock trading system might require more precision – in this application, a few percent error could make the difference between retiring in luxury in Hawaii and retiring in a cardboard box under a bridge.

One factor controlling the accuracy to which we can estimate probabilities in this kind of 'coin-toss' experiment is the number of replications conducted. At a fundamental level, the number of replications controls the resolution to which you can measure probability. For example, consider a 50-50 coin-toss experiment. If you toss the coin 10 times, you will get probability estimates with a resolution of 0.1 (0.0, 0.1, 0.2...). To be able to resolve probability down to the 0.001 level requires tallying the results of 1000 or more tosses – this is a simple consequence of the arithmetic used to turn a heads/tails tally into a probability estaimate.

Another less intuitive aspect of accuracy is the natural variation in a collection of random events. Continuing with our coin-toss experiment, if you were to toss an unbiased coin (truly 50-50 changes of heads or tails) 100 times, you would be unlikely to see exactly 50 heads and 50 tails appear – although the actual results would likely be 'close', and will tend to get even closer if you performed more replications.

In the birthday problem simulation model, you may have noticed that we chose 10,000 trials, which may seem to be an awfully large number. This was chosen for two reasons. First, it provides the ability to resolve probability to 1 part in 10,000 (0.0001 or 0.01%), which is far below the level of a percent or two which may be of interest in this application. Secondly, we are not flipping coins manually, and running 10,000 iterations of this experiment is not particularly difficult. Figure X.5 shows the results of running the simulation model with both 100 trials, and 10,000 trials, overlaid on the analytic (exact) model. In the 100 trials case, you can see quite a bit (several percent) of error, while the 10,000 trials case is almost visually indistinguishable from the analytic model when plotted.

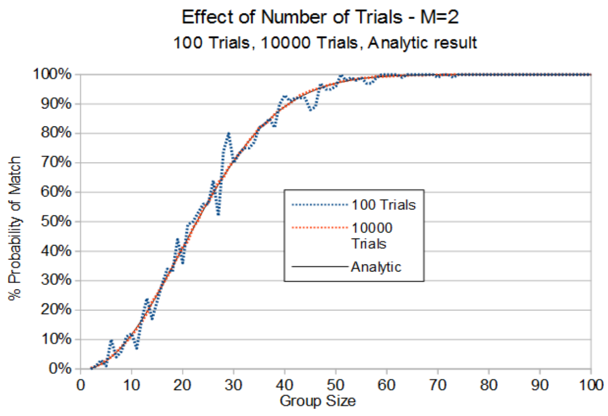

Figure X.5– 100 and 10,000 Trials vs. Analytic Results

Now that we have a model that seems to work, and have some idea of how to control error to acceptable limits, let's run it to find the probabilities of three people sharing a common birthday (M=3). Figure X.6 is a plot of the probabilities for N=3 to 100.

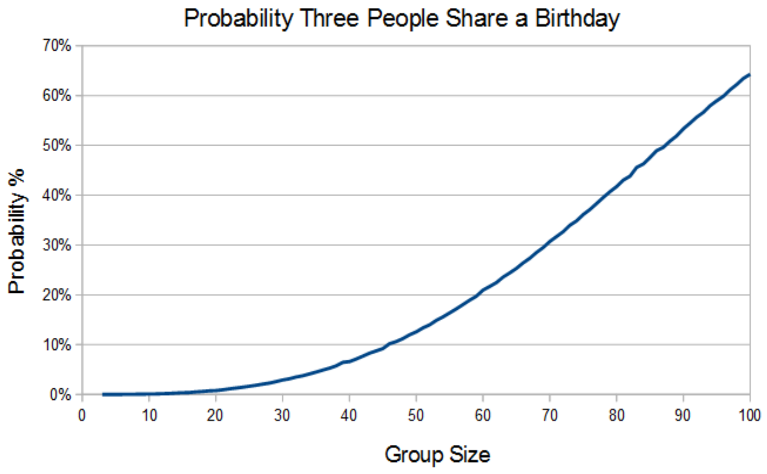

Figure X.6– Probability of three People Sharing a Birthday vs. Groupsize (N)

Using our 'precision calibrated eyeball', it appears that the group size needed to get a 50% probability of three coincident birthdays is in the mid 80's. Table X.2 shows numerical detail for this region of the graph.

N	Mean	StDev
80	41.73%	0.50%
81	43.02%	0.52%
82	43.81%	0.51%
83	45.58%	0.84%
84	46.26%	0.56%
85	47.55%	0.47%
86	48.94%	1.00%
87	49.61%	0.38%
88	50.87%	0.72%
89	51.93%	0.47%
90	53.34%	0.42%

Table X.2 Probability of Three People Sharing Birthday, N=80...90

Note that in this case, we ran the simulation twenty times, so as to be able to obtain a mean value, which one might expect to have a lower error than that of individual runs (hopefully!). Multiple runs also allows us to obtain standard deviation, which provides an indication of how much the individual simulations vary from the mean. From these results, it is difficult to say exactly which group size provides a 50% chance of having three coincident birthdays, but it would appear that it is likely to be between 86 and 89. for purposes of impressing an audience, this is probably a 'good enough' result – especially since it is a smaller number than one might initially guess (not counting leap years, it takes 731 people to *guarantee* a three-way match – 365 x 2 + 1).

To conclude, this essay has demonstrated how a simple and easy-to-create simulation model can be used to obtain results that otherwise would require complex analysis of considerable sophistication. The downside of the simulation model is that it doesn't provide exact answers, but for applications such as the one explored here, approximate results with known error may be 'good enough' for most practical purposes.

Problems:

1) As previously noted, births are not uniformly distributed over the year. What modifications would need be made to the model to account for different likelihoods of births from month-to-month?
2) We saw that the statistical error of the simulation is related to the number of trials considered. One approach to reducing error is to keep increasing the number of trials and and then statatistically analyze the results of multiple runs, repeating until the desired error is achieved. For a binomial experiment, however, such as a coin toss or the result of the ***RunExperiment*** function the standard deviation (σ_P) of the estimated probability (P) can be estimated [4] by the expression

$$\sigma_P = \sqrt{\frac{P \times (1-P)}{Trials}}$$

Can you create a simulation that will calculate the probability with just enough trials to achieve a given standard deviation?

References:

[1] Scientific American, *Bring Science Home - Probability and the Birthday Paradox* , March 29, 2012, https://www.scientificamerican.com/article/bring-science-home-probability-birthday-paradox/, retrieved 2018-11-16

[2] UK Office for National Statistics, *How Popular is Your Birthday,* https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/livebirths/articles/howpopularisyourbirthday/2015-12-18, retrieved 2018-11-16

[3] Wolfram Mathworld, *Birthday Problem*, http://mathworld.wolfram.com/BirthdayProblem.html

retrieved 11/24/2018

[4] Jay L. Devore, *Probability and Statistics for Engineering and the Sciences 7th ed.*, Thompson Brooks/Cole, Belmont CA 2008, pp.568-576.