Simulating the birthday paradox in Scratch

You might have used the “pick random” coding block in Scratch when creating mathematical activities that required some variability or random elements. We can use this block instead of 12 cards to perform our birth-month experiment.

We can set it to pick from 1 to 12, and then just clicking it in the coding window produces a number. We can also assign this to a variable.

In order to draw 5 numbers, the easiest way is to use a list and a little loop.

Now this is already much quicker than drawing 5 cards from a deck – but at the moment, we need to work out ourselves whether there’s a match or not. So the next step is including some code that can look at this list and tell me if there are two numbers that match.

That’s not so straightforward – and working out *how* requires a good understanding of what functions are available in Scratch (or whichever coding language you’re using).

Now we could create a loop that takes each number and compares it to every other number (this might remind you of the handshake problem), but the reason why I won’t do it that way is because I can see that Scratch has a block that just checks if a list contains a particular thing.

So once I have my list, I can ask if it has the number 5, for example. So I want the algorithm to check, first whether there’s any duplicates of the first number, then if there are any duplicates of the second number and so on.

But clearly It wouldn’t make sense to ask while that number is still in the list, so what I do instead, is remove it from the list, keep a record of it, and then check if it’s still there. If it is, that means a success. If not, then I’ll move on to the next number until there are no numbers left.

So now I have an algorithm that automatically simulates drawing 5 cards, and then tells me if there’s a match or not. What’s next? Repeating the experiment over and over again!

So now I can perform the experiment 10 times, 100 times, 100000 times, almost instantly. And what I should find, is that this probability becomes fairly close to what the theoretical probability is. So we have an experimental probability, or empirical probability, obtained by looking out our success fraction - and then a theoretical probability, which is obtained using tree diagrams and multiplication.

And what’s more, it’s not that hard to extend this algorithm to the birthday problem. What’s different? I I can change the number of people in the list to 23, and then each of these needs to draw from 1 to 365 because I’m simulating a day of the year rather than a month. Then we have a nice algorithm that can perform thousands of experiments to estimate the probability, but we can also use it if we have a theoretical probability and we want to verify that it seems right.