Simulating sequential events in Scratch

One way to perform simulations in Scratch is to use if-then rules along with the random number generator.

Whenever we can consider a scenario as a set of sequential events, we can model the possibilities along with their likelihood using a tree diagram. Each set of branches essentially represents an if-then and else statement, with the first outcome being obtained with a given probability, and the other outcome otherwise. Even if we have three possible outcomes, we can set up separate if-statements, or sometimes we can nest multiple if-else statements inside one another. And now we just keep in mind that the condition that will lead to either set of actions needs to be modelled with randomness.

Let's return to our trick cards game. What are the events that take place here?

First one of the three cards is selected randomly.

Then, one of the sides of the selected card would be shown, also randomly, although it won't make a difference for the two trick cards. And we're asked:

Given that we were shown a card-back – what is the probability that the other side will show the two of hearts.

So out of the situations showing a card back, what fraction of them end up showing the 2, and calculating this gives us a third – so that's the theoretical probability. How could we model it in Scratch so that we're able to run a large number of simulations that could give some support to this theoretical calculation?

We'll achieve this in the following way. First, we'll set up a variable so that we can keep track of which random card was chosen. We assign a random number between 1 and 3 to this, corresponding with which card is selected.

Then we can define a set of actions depending on what the value of that random card is.

We'll keep track of the card shown using a variable as well – we could easily assign the words "back" and "front" here, but we'll just use the numbers 1 and 2, 1 to indicate a card-back, and 2 indicates the 2 of hearts.

Now if it was the first card, the card shown is always a card back.

Then the second card always shows a 2 of hearts. It's only in the case of the third card that we need to model the randomness of choosing one side of the card or the other. We can do this using the pick random block again – this time, we'll just say that if the random number is 1, then we'll take one action, and otherwise we'll follow the actions in the else section. Note that we're not assigning this random number to a variable, although we do change the value of card shown depending on which random number, 1 or 2, was chosen.

So this set of instructions simulates our scenario, and each time, we get told what the random card is, and what was shown. We could work out what was on the other side based on this information.

So that we can run 10s or 1000s of simulations, we'll now put this all inside a loop and for this we'll need to set up some new variables so that we can calculate our overall results. To make things simple, we'll use a variable to say what was on the other side (we're not actually going to use this information, but it might help us later if we need to debug our code). Then for calculating our probability, we need to count all the times a card back was shown, and also the number of times that both a card back was shown AND the 2 of hearts was on the other side.

Make sure we reset these before we run our experiments.

And now finally, we can set up a calculation at the end that will give us our estimated probability.

Running this the first time with 100 simulations, we note that a card back was shown 51 times (which is about half of the time) and that 17 of those times, the other side was a heart. This is exactly our theoretical probability however I think we were a little lucky.

Changing the number of simulations we can see that we don't always get this close to the real probability, even when it is 100 like that first time. However once we set the number of simulations to ten thousand, we can see that the estimated probability is always around that 32 – 34 percent mark – which is very close to what the theoretical calculation tells us.