More Excel Dice

Making a VBA routine to provide a range of non-repeating, non-volatile random numbers.

Some time ago, I wrote an article called "Throwing Dice with Excel". In that article (available for download at http://www.grbps.com/articles) I looked at how to generate random numbers using Excel's built-in functions and how to use copy and paste special when non-volatile random numbers are needed.

Just recently, a reader emailed to ask for help. He needed to generate four random numbers between 1 and 35. Fine; this doesn't represent a problem. The twist was, the four numbers had to be unique: there could be no numbers duplicated among the four. This raises a real problem, because the nature of random numbers is that they are random. We all know that even if the coin lands heads up ten times in a row, the chance of it doing so again on the next toss is still 50%: the coin has no memory. Similarly, throw the dice twice and there's a possibility you'll get the same result both times. What my correspondent really needed was a way of drawing numbers from a bag. Each time a number is drawn, that number ceases to be in the bag for the next draw.

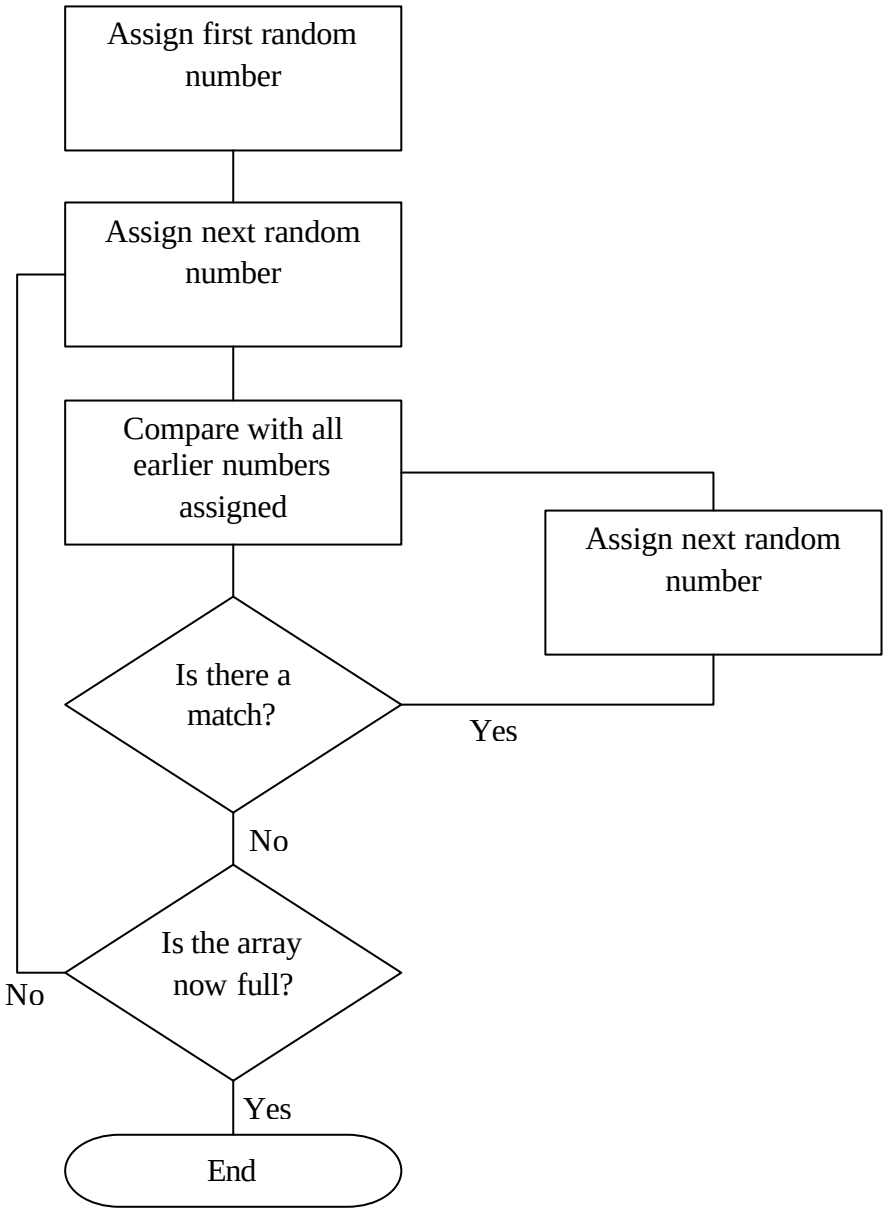
Now, I am a great believer in the power of Excel's built-in functions; I only resort to VBA code when I have to. Whilst it would probably be possible to create this 'number bag' with formulae, every approach I considered was very big and very complicated. Those are two adjectives I'll go a long way to avoid, and my instinct was that VBA would yield a far more straightforward solution this time.

Let's recap on what I needed to do. I needed four random numbers from a lowest possible value of 1 to a highest possible value of 35. In line with my normal practice, rather than hard-code these into my program, I assigned them as constants, which would make life a lot easier if I needed the code later for another situation. Here is my first section of code:

```
Sub GetRandomNumbers()

    Const intLowest As Integer = 1              'Lowest number required
    Const intHighest As Integer = 35            'Highest number required
    Const intHowMany As Integer = 4             'Number of different values to return

'Set up array to hold the random value
    Dim arrRandom(1 To intHowMany) As Integer

End Sub
```

Note that I've used constants rather than variables to hold my three main driving values. This allows me – for one thing – to use one of these values to define the dimensions of the array I'll use to hold the random numbers (I can only use a constant to do this, not a variable.)

My basic approach in this code will be as outlined in the flowchart below:



One of the dangers of using a loop structure is the possibility of the code locking up in an ‘infinite loop’, where the exit condition is never met. In my loop, this could happen if the number of different values required is bigger than the range of allowed numbers; there aren’t 10 whole numbers between 1 and 5, for example.

A little code before we get into the loop will take care of this:

```
'Check to prevent infinite loop
    If intHowMany > (intHighest - intLowest + 1) Then
        MsgBox "Too many numbers or too small a range."
        Exit Sub
    End If
```

And now we can put the loop into operation. Here is how we assign the first random number:

```
'Generate the first random value
    arrRandom(1) = Int((intHighest + 1 - intLowest) * rnd) + intLowest
```

If you've read the 'Throwing Dice' article, you'll recognise this as the VBA equivalent of the RAND formula I outlined there.

Of course, the first random number cannot possibly be a duplicate, so there is nothing to check it against. From this point in, we're in the main loop. As part of the loop, I'll need to be able to set a flag, so the first thing I'll do is declare a variable for this purpose:

```
    Dim booFlag As Boolean
```

Here is the main loop:

```
'Generate subsequent random values and ensure no dupes
    For i = 2 To intHowMany
        Do
            booFlag = False
            arrRandom(i) = Int((intHighest + 1 - intLowest) * rnd) + intLowest
            For j = 1 To i - 1
                If arrRandom(j) = arrRandom(i) Then booFlag = True
            Next j
        Loop Until booFlag = False          'Ensures value not kept if it is a dupe
    Next i
```

Note that this is really three loops:

(i) The For/Next 'i' loop which cycles through the second and subsequent random numbers required

(ii) The For/Next 'j' loop which, once a random number is provisionally assigned, cycles through each earlier number to check it is not a duplicate. If it is a duplicate, the flag variable booFlag is set to True.

(iii) The Do/While loop which prevents progress if booFlag has been set to True, sending the program back to choose a new number instead and test again.

By the time the code run to the end of this section I will have an array full of random numbers with no duplicates. What I now need to do is to report these numbers back to the worksheet. This is a job for another loop:

```
'Write the values back to the worksheet - change "B1" reference as required
    For i = 1 To intHowMany
        Range("B1").Offset(i, 0).Value = arrRandom(i)
    Next i
```

There is just one more addition to make before the code is complete. In VBA, the Rnd function is only really 'pseudo-random' since it works from a seed based on the last result obtained. If I want to produce the most random result, I need to include the following command before first using Rnd:

```
Randomize                       'Resets random seed
```

This will ensure the random seed is reset, based on the system timer.

The full code looks like this:

```
Sub GetRandomNumbers()

    Const intLowest As Integer = 1              'Lowest number required
    Const intHighest As Integer = 35            'Highest number required
    Const intHowMany As Integer = 4             'Number of different values to return

    Dim booFlag As Boolean

'Check to prevent infinite loop
    If intHowMany > (intHighest - intLowest + 1) Then
        MsgBox "Too many numbers or too small a range."
        Exit Sub
    End If

'Set up array to hold the random value
    Dim arrRandom(1 To intHowMany) As Integer

    Randomize                       'Resets random seed

'Generate the first random value
    arrRandom(1) = Int((intHighest + 1 - intLowest) * Rnd) + intLowest

'Generate subsequent random values and ensure no dupes
    For i = 2 To intHowMany
        Do
            booFlag = False
            arrRandom(i) = Int((intHighest + 1 - intLowest) * Rnd) + intLowest
            For j = 1 To i - 1
                If arrRandom(j) = arrRandom(i) Then booFlag = True
            Next j
        Loop Until booFlag = False          'Ensures value not kept if it is a dupe
    Next i

'Write the values back to the worksheet - change "B1" reference as required
    For i = 1 To intHowMany
        Range("B1").Offset(i, 0).Value = arrRandom(i)
    Next i

End Sub
```

The only thing I now have to put into place is a mechanism to call this code, since calling the macros dialog each time will quickly become tiresome. Which method I choose will depend on what I need the numbers to do, and this is one aspect of my correspondent's needs about which I know nothing.

One method would be to create a button on the worksheet or on a custom toolbar to run the code. Another would be to call if from the Open event of the workbook, to force it to get new random numbers whenever it is opened. Or, to use it like a volatile function, it could run from the Change event of the Worksheet.

So, we've built a VBA program which will return a given number of random numbers within a given range without any duplicates. Rather than dice game lovers, this will be of most interest, I suppose, to bingo aficionados!