Subject: Re: Randomize array order
Posted by James Kuyper on Fri, 27 Jul 2007 16:29:49 GMT
View Forum Message <> Reply to Message

```
Allan Whiteford wrote:
>  David Streutker wrote:
>>  On Jul 26, 2:32 pm, hradilv <hrad...@yahoo.com> wrote:
>>
>>> On Jul 26, 11:57 am, David Streutker <dstreut...@gmail.com> wrote:
>>>
>>>
>>>
>>>
>>>> How about a Knuth shuffle?
>>>
>>>> (Disclaimer: I'm not a statistician; I just found it on Wikipedia.)
>>>
>>>> function kunsort, array
>>>>   na = n_elements(array)
>>>>   random = randomu(seed,na) * (na - array - 1) + array
>>>>   for i=0L,na-2 do array[i] = array[random[i]]
>>>>   return, array
>>>> end
>>>
>>>> Added to Vince's code (last column):
>>>
>>>>          0     100000    0.0619998    0.0780001    0.0320001
>>>>          1     166810     0.125000     0.125000    0.0780001
>>>>          2     278256     0.282000     0.218000     0.141000
>>>>          3     464158     0.547000     0.406000     0.235000
>>>>          4     774263      1.07800     0.657000     0.390000
>>>>          5    1291549      1.93700      1.09400     0.703000
>>>>          6    2154435      3.51500      1.89100      1.20300
>>>>          7    3593812      6.34400      3.11000      1.98400
>>>>          8    5994841      11.5470      5.21800      3.36000
>>>>          9   10000000      20.3750      8.67200      5.60900
>>>
>>>> Windows XP, dual 2.66 GHz, 3 GB RAM, IDL 6.3
>>>
>>> I'm not sure, but I think that will give you "with replacement".
>>
>>
>>  You're right, I wasn't swapping.  Corrected, and in the form of
>>  Allan's method:
>>
>>  function kunsort, array
>>    na = n_elements(array)
>>    rarray = array
```

```
>>
>>    b = randomu(seed,na-1) * (na - lindgen(na-1) - 1) + lindgen(na-1)
>>    for i=0L,na-2 do begin
>>      tmp = rarray[i]
>>      rarray[i] = rarray[b[i]]
>>      rarray[b[i]] = tmp
>>    endfor
>>
>>    return, rarray
>>  end
>>
```

>> With the change, it's slightly slower than Allan's. However, for what
>> it's worth, there are claims this is a less biased method. (Again, I
>> am no expert. But the recent poker craze seems to have revived
>> interest in the probabilities of shuffling.)
>>
>
> Yours is doing a bit more than mine was in that it's creating a copy of
> the original data rather than in-place swapping so that would make yours
> a bit slower (but probably more useful). You can probably also get a
> speed up by converting "b" to a long at creation time rather than
> implicitly ever time you use it.
>
> Looking over what a Knuth Shuffle is supposed to to, it seems that
> you're only supposed to swap with an element you've not already passed
> over; my code didn't do this but yours does. I guess Knuth is smarter
> than me :). Although, I tend to not believe anything which appears on
> Wikipedia.
>
> However, in your code, it looks like the "na-1" in the creation of "b"
> and the "na-2" in the loop will mean that the last element of the array
> never gets swapped.

As I understand it, I think that neither program correctly implements
Knuth's algorithm. Here's my (minimally tested) attempt. I wrote it as
an in-place shuffle, to save space:

```
PRO knuth_shuffle, array
   na = N_ELEMENTS(array)
   b = LONG((na+2-LINDGEN(na-1))*RANDOMU(seed, na-1))
   FOR i=na-1, 1, -1 DO BEGIN
      temp = array[b[i-1]]
      array[b[i-1]] = array[i]
      array[i] = temp
   ENDFOR
END
```