Subject: Re: fun with random numbers
Posted by John-David T. Smith on Thu, 02 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Struan Gray wrote:
>
> David Fanning, davidf@dfanning.com writes:
>
>> morisset@my-deja.com (morisset@my-deja.com) writes:
>>
>>> Or you might use common block (Ok, David, I know...) or
>>> a system variable:
>>
>> Or even, possibly, a pointer. :-)
>
> Singleton object.

Doubly-recursive self-referential two-headed list object.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

Subject: Re: fun with random numbers
Posted by Struan Gray on Thu, 02 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning, davidf@dfanning.com writes:

> morisset@my-deja.com (morisset@my-deja.com) writes:
>
>> Or you might use common block (Ok, David, I know...) or
>> a system variable:
>
> Or even, possibly, a pointer. :-)

Singleton object.

Struan

Subject: Re: fun with random numbers
Posted by morisset on Thu, 02 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

David wrote:
> Or even, possibly, a pointer. :-)

I don't see how to do with a pointer...
The goal being not to add a parameter to the calling
sequence of test.pro or you will have to check every
routine using every routine using eve... using test
'till you are sure not to loose the value in any
calling sequence.

Or I miss one more time something with the pointers? ;-)

Cheers,
Christophe.

Subject: Re: fun with random numbers
Posted by davidf on Thu, 02 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

morisset@my-deja.com (morisset@my-deja.com) writes:

> Or you might use common block (Ok, David, I know...) or
> a system variable:

Or even, possibly, a pointer. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: fun with random numbers
Posted by morisset on Thu, 02 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Hi,
The bug comes from the unidentification of seed in
the first randomu call of every call of test.pro
The solution is to save the value of seed between
different calls of test.pro
Or you might use common block (Ok, David, I know...) or
a system variable:

```
PRO test
common seed_for_test,seed

  x= RANDOMU(seed,1)
  y= RANDOMU(seed,4)
  PRINT,'x',x
  PRINT,'y',y

  RETURN

END
```

or

```
PRO test
   defsysv,'!seed',EXISTS = seed_exists
   if seed_exists then seed = !seed

   x= RANDOMU(seed,1)
   y= RANDOMU(seed,4)

   defsysv,'!seed',seed

   PRINT,'x',x
   PRINT,'y',y

   RETURN

END
```

Sent via Deja.com http://www.deja.com/

Before you buy.

Subject: Re: fun with random numbers
Posted by landsman on Thu, 02 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

In article <38BD4DB6.E9F60A17@phys.ucalgary.ca>,
Brian Jackel <bjackel@phys.ucalgary.ca> wrote:
> Greetings
>
> We've just spent a couple hours figuring out why a Monte-Carlo
> simulation was giving peculiar (ie. wrong) results.
>
> The "test" procedure creates two random variables and prints
> them. Called twice, you might think that the results would be
> totally different. Here's an example result:
>
> x 0.120838
> y 0.649213 0.577647 0.139354 0.745442
>
> x 0.649213
> y 0.577647 0.139354 0.745442 0.942317
>
showing how the "random" numbers are merely being shifted by one
position.

Sigh.....

This bug has existed since IDL V5.1.1 and continues into IDL V5.3. It turns out that there were *two* RANDOMU bugs introduced into V5.1.1. The first one was that the SEED variable was being initialized to the same value at the start of each session. This bug was fixed in V5.2.1. But the bug described above by Brian Jackel appears to continue into V5.3.

Below I update my epic on the history of RANDOMU problems. Note that I also give the wrapper RANDOM() routine suggested by Pat Broos to fix the problem with multiple RANDOMU calls described above.

--Wayne Landsman landsman@mpb.gsfc.nasa.gov

****

V4.0.1: No problems? (But the algorithm was rumored to be far from state of the art.)

V5.0: RANDOMU could yield a non-random distribution if two programs

using RANDOMU are interleaved. For example, in the program demo.pro given at the bottom of this message, the command demo,/breakit will show a significant excess in the distribution of random numbers between 0 and 0.03.

V5.1: A *negative* seed value must be specified if you want to preserve
the same "random" sequence

```
IDL> seed = 2 & print, randomu(seed, 3)
     0.0594004    0.982075    0.358593
IDL> seed = 2 & print, randomu(seed, 3)
     0.831999    0.303037    0.506712
```

but

```
IDL>  seed = -2 & print, randomu(seed, 3)
       0.342299    0.402381    0.307838
IDL> seed = -2 & print, randomu(seed, 3)
       0.342299    0.402381    0.307838
```

This isn't necessarily a bug, but it means that RANDOMU works differently in V5.1 than in all other IDL versions.

V5.1.1 and V5.2:

(1) The seed variable is now initialized to the same value at the start of each session rather than the system clock. Thus, Monte Carlo simulations from different IDL sessions, might yield decidedly unrandom results.

(2) Perhaps more insidious, only the first call to RANDOMU is initialized inside a program. Thus, if one calls the following program test.pro multiple times, you will see that the "random" vector is simply the vector on the previous call, shifted by one.

```
PRO test
print, randomu(seed)
print, randomu(seed,6)
return
end
```

V5.2.1 and V5.3

Problem (1) in V5.1.1 has been fixed -- the seed variable is correctly initialized. But problem (2) concerning multiple RANDOMU calls inside a program remains. For this last problem, Pat Broos suggests using the

following wrapper program to RANDOMU to store the seed value in a common block.

```
 FUNCTION random, n1, n2, n3, NEW_SEED=new_seed, _EXTRA=extra

 COMMON random_seed, seed

 if keyword_set(new_seed) then seed = long(new_seed)

 case n_params() of
    0: return, randomu(seed,          _EXTRA=extra)
    1: return, randomu(seed,n1,       _EXTRA=extra)
    2: return, randomu(seed,n1,n2,    _EXTRA=extra)
    3: return, randomu(seed,n1,n2,n3, _EXTRA=extra)
 endcase

 end


 ************************************************************ ************

 FUNCTION lib_random

 return, randomu(other_seed,1)
 end


 PRO demo, x, BREAK_IT=break_it

 ; Type demo,/breakit to see the "non-random" distribution that can
result in ;
 V5.0.  Works correctly in earlier and later IDL versions

 x = fltarr(100000)

 for ii = 0L, n_elements(x)-1 do begin
   x(ii) = randomu(seed,1)

   if keyword_set(break_it) then dummy = lib_random()
 endfor

 h = histogram( x, MIN=0.0, BIN=0.01 )
 plot, h, PSYM=10
 print, h
 return
 end
```