Subject: random number trap
Posted by peter on Tue, 29 Aug 1995 07:00:00 GMT
View Forum Message <> Reply to Message

A warning about the random number generator in IDL/PV-Wave (not a bug, per se, but something to watch out for).

As the documentation states, if the seed value given to randomu is undefined, it is derived from the system time. The time only changes once per second, however. So if you repeatedly call a procedure that calls randomu, the return will be the same if the calls occur within a second of each other, but will be different if they are in different seconds.

This can lead to random numbers being a *lot* more structured than you expect. I had naively expected that the seed value would change each microsecond, so this behavior came as a bit of a surprise.

The solution is to place the seed variable in a common block so that it is preserved from call to call, and then each returned sequence will truly be random.

Peter

Peter Webb, HP Labs Medical Dept
E-Mail: peter_webb@hpl.hp.com
Phone: (415) 813-3756

Subject: Re: random number trap
Posted by Adam Shane on Wed, 27 Sep 1995 07:00:00 GMT
View Forum Message <> Reply to Message

scott@abyss.ATMOS.ColoState.Edu (Scott Denning) wrote:
>
> Peter Webb writes
>> A warning about the random number generator in IDL/PV-Wave (not a bug,
>> per se, but something to watch out for).
>>
>> As the documentation states, if the seed value given to randomu is
>> undefined, it is derived from the system time. The time only changes
>> once per second, however. So if you repeatedly call a procedure that
>> calls randomu, the return will be the same if the calls occur within a
>> second of each other, but will be different if they are in different
>> seconds.
>>
>> This can lead to random numbers being a *lot* more structured than you

```
>>  expect.  I had naively expected that the seed value would change each
>>  microsecond, so this behavior came as a bit of a surprise.
>>
>>  The solution is to place the seed variable in a common block so that it
>>  is preserved from call to call, and then each returned sequence will
>>  truly be random.
>>
>>  Peter
>>
>>  -------------------------------
>>  Peter Webb, HP Labs Medical Dept
>>  E-Mail: peter_webb@hpl.hp.com
>>  Phone: (415) 813-3756
>
>  Actually, the problem goes much deeper than the granularity of the system
>  time, and hinges on what you mean by "random." Many scientific users
>  expect a "random" variable to have a Gaussian distribution, which no
>  "random number generator" in any language is likely to provide.
>
>  For an excellent discussion of this problem, as well as nice, simple
>  solutions, see W. H. Press et al., 1992: Numerical Recipes, Cambridge
>  University Press,  Chapter 7 (Random Numbers).
>
>  --
>  A. Scott Denning                    scott@abyss.Atmos.ColoState.edu
>  Dept. of Atmospheric Science                   Phone (970)491-2134
>  Colorado State University                      Fax1 (970)491-8428
>  Fort Collins, CO 80523-1371                    Fax2 (970)491-8449
```

In IDL there are two routines, RANDOMU and RANDOMN. The former gives
a set of random numbers of UNIFORM distribution over the interval
[0,1) (assuming you don't keep changing the random number seed as
stated above). The latter will give a NORMAL or GAUSSIAN distribution
that you say you "expect." A simple but informative example was
supplied in the IDL documentation for version 3.6.

Adam Shane
Arete Associates

Subject: Re: random number trap
Posted by James Theiler on Wed, 04 Oct 1995 07:00:00 GMT
View Forum Message <> Reply to Message

Peter Webb writes

```
>>  A warning about the random number generator in IDL/PV-Wave (not a bug,
>>  per se, but something to watch out for).
>>
```

>> As the documentation states, if the seed value given to randomu is
>> undefined, it is derived from the system time. The time only changes
>> once per second, however. So if you repeatedly call a procedure that
>> calls randomu, the return will be the same if the calls occur within a
>> second of each other, but will be different if they are in different
>> seconds.
>>
>> This can lead to random numbers being a *lot* more structured than you
>> expect. I had naively expected that the seed value would change each
>> microsecond, so this behavior came as a bit of a surprise.
>>
>> The solution is to place the seed variable in a common block so that it
>> is preserved from call to call, and then each returned sequence will
>> truly be random.
>>
>> Peter
>>
>> --------------------------------
>> Peter Webb, HP Labs Medical Dept
>> E-Mail: peter_webb@hpl.hp.com
>> Phone: (415) 813-3756

In fact, this points to another trap (again, not a bug per se) with the
random number generator. If you provide the seed yourself, instead of
depending on the system clock (eg, so that you can repeat a monte-carlo
experiment) you should beware of 1) using small seeds (leads to small
random numbers), and 2) using seeds that are nearly equal (leads to
random numbers that are nearly equal). For instance,

```
IDL> seed=1 & print,randomu(seed)
  7.82637e-06
IDL> seed=2 & print,randomu(seed)
  1.56527e-05
IDL> seed=100000 & print,randomu(seed)
     0.782637
IDL> seed=100001 & print,randomu(seed)
     0.782645
```

The problem is that the random number is directly proportional to the
seed (modulo some large number), so if you want to make independent
runs, you need to choose "random" seeds.   One workaround is to have
a few "junk=randomu(seed)" statements near the beginning of your program.
In this example, even with only one such statement, the seeds have
sufficiently diverged that the random numbers are not so nearly equal.

```
IDL>  seed=100000 & junk=randomu(seed) & print,randomu(seed)
     0.778814
IDL>  seed=100001 & junk=randomu(seed) & print,randomu(seed)
```

```
    0.910352
```

By the way, I think one should set the seed only once in a run (or let
the system clock set it), and *not* reset it (or let the clock reset it)
for the rest of the run. Sounds like that is what Peter's common block
is doing. That way, even the above problem will only apply to the first
few random numbers that are generated in a run.

Note, this applies to the gaussian randomn() function as well.

jt
--
James Theiler, MS-D436, LANL, Los Alamos NM 87545; jt@lanl.gov