Subject: fun with random numbers
Posted by Brian Jackel on Wed, 01 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

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
```

The problem is that for the first random number, "seed" is undefined, so a generic state is used. From the docs:

"In addition to states maintained by the user in variables, the RANDOMU and RANDOMN functions contain a single shared generic state that is used if a named variable is not supplied as the Seed argument or the named variable supplied is undefined. The generic state is initialized once using the time-of-day, and may be re-initialized by providing a Seed argument that is a constant or expression."

Which is all fine. This generic state is used to produce the first random number, the seed is updated, then apparently used to overwrite the generic state. Another four random numbers are produced, with the seed changing every time. However, none of these values are used to overwrite the generic state.

Consequently, on the next call, we start with a generic state that is only advanced *one* from the previous call. Hence the sequence of "y" values starts at the second position of the previous "y" sequence.

I would assert that this is not good behaviour. The "generic state" should
1) always be overwritten with the last seed OR
2) should never be changed without an explicit call by the user

Any comments?

```
 ============================================================ ===
PRO test

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

  RETURN

END

test
test

END
```