Subject: Re: Randomu seed initialization
Posted by penteado on Wed, 11 Nov 2009 20:22:35 GMT
View Forum Message <> Reply to Message

On Nov 11, 5:54 pm, Conor <cmanc...@gmail.com> wrote:
> So I'm really wondering if I've hit some sort of strange IDL bug
> (although everytime I thought that so far I've been wrong). Still,
> thought I'd post about it.
>
> I've got a routine I'm running to do some simulations. It calls a
> function (generate_population) which generates 1000 variables
> populated along a certain distribution. In my routine I later
> generate another set of 1000 variables to select some of the generated
> population. In both of these cases IDL has to initialize the random
> seed generator because (of course) the generate_population routine is
> a separate function and so the main routine can't use the 'seed'
> variable generated in generate_population.
>
> However, I noticed that the cut being made was very strange and not at
> all what it should have been. After some examination, I discovered
> that the random variables being created by generate_population were
> the exact same random variables being used to make the cuts - two
> different calls to randomu() were returning exactly the same random
> variables. Which means that they were using the same seed. Except I
> wasn't giving either one the seed - it was being generated
> automatically by IDL. Weird... I've tried reproducing this by making
> simple routines that do the same sort of thing, but I don't have the
> same problem. Which would make me think the problem is with something
> I'm doing, but at the same time I just don't see any problems in my
> code - the part dealing with the random number generators is simple
> enough. I'm hoping someone here might have some insights into how IDL
> initializes the random number generator that might help me figure out
> what's going on. For now I've fixed the problem because I pass the
> seed back and forth between the two routines (so it only gets
> initialized once), but I'd really like to know what's happening. It's
> hard for me to post the code because it's part of a routine that relys
> on some large data files, but if people think it might help I might be
> able to parse it down to something post-able.

What exactly do you mean by "wasn't giving either one the seed"?
Randomu cannot be called without a seed argument, so what were you
using?

Did you see randomu's help for the explanation of how the seed
variable works, in all the different ways it can assume?

Subject: Re: Randomu seed initialization
Posted by Conor on Thu, 12 Nov 2009 13:59:44 GMT
View Forum Message <> Reply to Message

On Nov 11, 3:22 pm, pp <pp.pente...@gmail.com> wrote:
> On Nov 11, 5:54 pm, Conor <cmanc...@gmail.com> wrote:
>
>
>
>> So I'm really wondering if I've hit some sort of strange IDL bug
>> (although everytime I thought that so far I've been wrong). Still,
>> thought I'd post about it.
>
>> I've got a routine I'm running to do some simulations. It calls a
>> function (generate_population) which generates 1000 variables
>> populated along a certain distribution. In my routine I later
>> generate another set of 1000 variables to select some of the generated
>> population. In both of these cases IDL has to initialize the random
>> seed generator because (of course) the generate_population routine is
>> a separate function and so the main routine can't use the 'seed'
>> variable generated in generate_population.
>
>> However, I noticed that the cut being made was very strange and not at
>> all what it should have been. After some examination, I discovered
>> that the random variables being created by generate_population were
>> the exact same random variables being used to make the cuts - two
>> different calls to randomu() were returning exactly the same random
>> variables. Which means that they were using the same seed. Except I
>> wasn't giving either one the seed - it was being generated
>> automatically by IDL. Weird... I've tried reproducing this by making
>> simple routines that do the same sort of thing, but I don't have the
>> same problem. Which would make me think the problem is with something
>> I'm doing, but at the same time I just don't see any problems in my
>> code - the part dealing with the random number generators is simple
>> enough. I'm hoping someone here might have some insights into how IDL
>> initializes the random number generator that might help me figure out
>> what's going on. For now I've fixed the problem because I pass the
>> seed back and forth between the two routines (so it only gets
>> initialized once), but I'd really like to know what's happening. It's
>> hard for me to post the code because it's part of a routine that relys
>> on some large data files, but if people think it might help I might be
>> able to parse it down to something post-able.
>
> What exactly do you mean by "wasn't giving either one the seed"?
> Randomu cannot be called without a seed argument, so what were you
> using?
>
> Did you see randomu's help for the explanation of how the seed
> variable works, in all the different ways it can assume?

Sorry. By "didn't give it a seed" I mean, "passed it an undefined variable", under which circumstances it initializes the seed itself. I've read through randomu's help on how the seed works a number of times, including yesterday when I had this problem. This is the first time I've ever had any problems using randomu.

Subject: Re: Randomu seed initialization
Posted by Jeremy Bailin on Thu, 12 Nov 2009 19:31:04 GMT
View Forum Message <> Reply to Message

On Nov 12, 8:59 am, Conor <cmanc...@gmail.com> wrote:
> On Nov 11, 3:22 pm, pp <pp.pente...@gmail.com> wrote:
>
>
>
>
>
>> On Nov 11, 5:54 pm, Conor <cmanc...@gmail.com> wrote:
>
>>> So I'm really wondering if I've hit some sort of strange IDL bug
>>> (although everytime I thought that so far I've been wrong). Still,
>>> thought I'd post about it.
>
>>> I've got a routine I'm running to do some simulations. It calls a
>>> function (generate_population) which generates 1000 variables
>>> populated along a certain distribution. In my routine I later
>>> generate another set of 1000 variables to select some of the generated
>>> population. In both of these cases IDL has to initialize the random
>>> seed generator because (of course) the generate_population routine is
>>> a separate function and so the main routine can't use the 'seed'
>>> variable generated in generate_population.
>
>>> However, I noticed that the cut being made was very strange and not at
>>> all what it should have been. After some examination, I discovered
>>> that the random variables being created by generate_population were
>>> the exact same random variables being used to make the cuts - two
>>> different calls to randomu() were returning exactly the same random
>>> variables. Which means that they were using the same seed. Except I
>>> wasn't giving either one the seed - it was being generated
>>> automatically by IDL. Weird... I've tried reproducing this by making
>>> simple routines that do the same sort of thing, but I don't have the
>>> same problem. Which would make me think the problem is with something
>>> I'm doing, but at the same time I just don't see any problems in my
>>> code - the part dealing with the random number generators is simple
>>> enough. I'm hoping someone here might have some insights into how IDL
>>> initializes the random number generator that might help me figure out

>>> what's going on. For now I've fixed the problem because I pass the
>>> seed back and forth between the two routines (so it only gets
>>> initialized once), but I'd really like to know what's happening. It's
>>> hard for me to post the code because it's part of a routine that relys
>>> on some large data files, but if people think it might help I might be
>>> able to parse it down to something post-able.
>
>> What exactly do you mean by "wasn't giving either one the seed"?
>> Randomu cannot be called without a seed argument, so what were you
>> using?
>
>> Did you see randomu's help for the explanation of how the seed
>> variable works, in all the different ways it can assume?
>
> Sorry. By "didn't give it a seed" I mean, "passed it an undefined
> variable", under which circumstances it initializes the seed itself.
> I've read through randomu's help on how the seed works a number of
> times, including yesterday when I had this problem. This is the first
> time I've ever had any problems using randomu.

If I had to speculate, I'd say that the variable that you're passing
as the seed isn't undefined like you think it is. Have you tried doing
a "help, seed" right before the randomu call?

-Jeremy.

Subject: Re: Randomu seed initialization
Posted by Matt[2] on Thu, 12 Nov 2009 20:32:42 GMT
View Forum Message <> Reply to Message

Jeremy Bailin <astroconst@gmail.com> writes:

> If I had to speculate, I'd say that the variable that you're passing
> as the seed isn't undefined like you think it is. Have you tried doing
> a "help, seed" right before the randomu call?

If I was going to speculate, I would guess the undefined seed gets set
to the same value internally every time. Thereby starting the same
string of pseudo-random numbers.

I usually use:
seed = long(systime(/seconds))

To initialize my seed. It's not perfect, but it's a start.

Also remember from the docs: "Each independent random number sequence

should maintain its own state variable. To maintain a state over
repeated calls to a procedure, the seed variable may be stored in a
COMMON block."

Matt

--
Matthew Savoie - Scientific Programmer
National Snow and Ice Data Center
(303) 735-0785 http://nsidc.org

Subject: Re: Randomu seed initialization
Posted by Chris[6] on Thu, 12 Nov 2009 21:49:22 GMT
View Forum Message <> Reply to Message

On Nov 12, 10:32 am, sav...@nsidc.org wrote:
> Jeremy Bailin <astroco...@gmail.com> writes:
>> If I had to speculate, I'd say that the variable that you're passing
>> as the seed isn't undefined like you think it is. Have you tried doing
>> a "help, seed" right before the randomu call?
>
> If I was going to speculate, I would guess the undefined seed gets set
> to the same value internally every time. Thereby starting the same
> string of pseudo-random numbers.
>
> I usually use:
> seed = long(systime(/seconds))
>
> To initialize my seed. It's not perfect, but it's a start.
>
> Also remember from the docs: "Each independent random number sequence
> should maintain its own state variable. To maintain a state over
> repeated calls to a procedure, the seed variable may be stored in a
> COMMON block."
>
> Matt
>
> --
> Matthew Savoie - Scientific Programmer
> National Snow and Ice Data Center
> (303) 735-0785 http://nsidc.org

I've hit this issue before - calling RANDOMU with an undefined seed
many times produces collections of random numbers with very similar
properties. It's weird.

Successive calls to randomu which save the seed value avoid this
problem. Whenever I use randomu in a function, I do something like

```
function junk
common junk_seed, seed
x = randomu(seed)
return, x
end
```

which keeps track of the seed value automatically.

chris

Subject: Re: Randomu seed initialization
Posted by pgrigis on Thu, 12 Nov 2009 21:59:30 GMT
View Forum Message <> Reply to Message

As a general comment - anybody is wondering how useful
it is to have a poorly documented random number generator
of unknown robusteness and period in IDL?

I mean - all the docs say is that's "similar" to ran1
in numerical recipes - how similar? I mean - either
it is ran1 or it isn't :) if not, what? - and if it
is identical to ran1 - then we should know.

Also it's not clear to me what the seed really is -
seems overly complicated.

So I have had enough.

A while ago I decided to write my own random number
generator based on numerical recipes edition 3 - it's
much slower but very robust, and even though I
may not actually *need* such a good random number
generator, at least it's not a black box!

You can call it with a single integer seed or with 3 UL64's
- internally the seed is just 3 ulong64numbers - much
simpler then the seed from randomu :)

```
IDL> seed=17 & print,pg_ran(seed,10,/double)
    0.014634145      0.40536879      0.88335246      0.92373509
0.26807163     0.084861013      0.38462108      0.75499700
   0.0081408007      0.43871562
IDL> help,seed
SEED            ULONG64   = Array[3]
```

See header documentation for more info.

http://hea-www.cfa.harvard.edu/~pgrigis/idl_stuff/pg_ran.pro

Ciao,
Paolo

On Nov 12, 4:49 pm, Chris <beaum...@ifa.hawaii.edu> wrote:
> On Nov 12, 10:32 am, sav...@nsidc.org wrote:
>
>
>
>> Jeremy Bailin <astroco...@gmail.com> writes:
>>> If I had to speculate, I'd say that the variable that you're passing
>>> as the seed isn't undefined like you think it is. Have you tried doing
>>> a "help, seed" right before the randomu call?
>
>> If I was going to speculate, I would guess the undefined seed gets set
>> to the same value internally every time. Thereby starting the same
>> string of pseudo-random numbers.
>
>> I usually use:
>> seed = long(systime(/seconds))
>
>> To initialize my seed. It's not perfect, but it's a start.
>
>> Also remember from the docs: "Each independent random number sequence
>> should maintain its own state variable. To maintain a state over
>> repeated calls to a procedure, the seed variable may be stored in a
>> COMMON block."
>
>> Matt
>
>> --
>> Matthew Savoie - Scientific Programmer
>> National Snow and Ice Data Center
>> (303) 735-0785 http://nsidc.org
>
> I've hit this issue before - calling RANDOMU with an undefined seed
> many times produces collections of random numbers with very similar
> properties. It's weird.
>
> Successive calls to randomu which save the seed value avoid this
> problem. Whenever I use randomu in a function, I do something like
>
> function junk

> common junk_seed, seed
> x = randomu(seed)
> return, x
> end
>
> which keeps track of the seed value automatically.
>
> chris

Subject: Re: Randomu seed initialization
Posted by Conor on Fri, 13 Nov 2009 16:31:49 GMT
View Forum Message <> Reply to Message

Thanks for the input guys:

Jeremy:

I've checked in my primary function, and the variable I pass is uninitialized at first (obviously it becomes initialized after the first call to randomu() and stays that way). I haven't specifically checked that it is uninitialized in the subroutine, but the subroutine is only about 5 lines long, and I'm certainly not initializing it. So if it is initialized, then that's certainly an IDL bug (and it seems unlikely to me).

Matt:

I checked this possiblity, by making a couple simple routines:

```
pro test
print,randomu(seed,1)
test2
end

pro test2
print,randomu(seed,1)
end
```

Run this and you will get two different answers out - i.e. the random seed is not initialized to the same value more than once. Which is why I don't understand what's going on with my routine: conceptually, this is the simple case of what I'm doing in my routine. This is why I really wish I knew what randomu() was doing. I'm wondering (and there are kinda hints to this in the documentation) if there is a hidden "global" seed, which is initialized by the first call to randomu (), and used by all subsequent calls to randomu() that have an uninitialized seed. Either that or the initialization of the seed is

quasi-random each time (which would be much better, IMO).

Chris:

Thanks for the suggestion, that's what I've done for now, and it fixes it. I just wish I understood what is going on.

Paolo: I could definitely go for some better documentation! Presumably if it was better documented I would be able to avoid problems like this, and I would (perhaps) be better able to determine what I was doing wrong, or if this is a bug. As near as I can determine I'm using randomu properly (it's not exactly a complicated function). To uninitialized variables go into two separate calls to randomu(), and the same random numbers come out both times. Is that supposed to happen ever?
