Subject: Re: Random selection
Posted by David Fanning on Sat, 11 Nov 2006 13:23:40 GMT
View Forum Message <> Reply to Message

Julio writes:

> I have a simple question (I guess...).
> I have an array with two columns and a thousand of rows. Each row has a
>
> pair of latitude and longitude.
>
>
> Array=floatarr(2, 1000)
>
>
> -27.3456 -54.6529
> -23.4546 -56.7263
> ... and so on...
>
>
> I need to retrieve only 100 pairs, randomly. How can I do that?

Like this:

```
  indices = Randomu(seed, 100) * 1000
  selected = array[*,indices]
```

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Random selection
Posted by greg michael on Sat, 11 Nov 2006 13:28:39 GMT
View Forum Message <> Reply to Message

Try:

```
 result=array[*,randomu(seed,100)*1000]
```

regards,
Greg

Subject: Re: Random selection
Posted by David Fanning on Sat, 11 Nov 2006 13:30:45 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> Like this:
>
> indices = Randomu(seed, 100) * 1000
> selected = array[*,indices]

Well, it probably should be

  indices = (Round(Randomu(seed, 100) * 1000) < 999

But you get the idea. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Random selection
Posted by Jean H. on Sat, 11 Nov 2006 20:45:29 GMT
View Forum Message <> Reply to Message

> Well, it probably should be
>
> indices = (Round(Randomu(seed, 100) * 1000) < 999
>
> But you get the idea. :-)
>
> Cheers,
>
> David

though the same index could be repeated... the "sort" method (cf Kenneth
 post) is really, I believe, the only one that return only different index!

Jean

Subject: Re: Random selection
Posted by David Fanning on Sat, 11 Nov 2006 21:08:42 GMT
View Forum Message <> Reply to Message

Jean H. writes:

> though the same index could be repeated... the "sort" method (cf Kenneth
> post) is really, I believe, the only one that return only different index!

I don't think so. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Random selection
Posted by Jean H. on Sat, 11 Nov 2006 21:25:54 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Jean H. writes:
>
>
>> though the same index could be repeated... the "sort" method (cf Kenneth
>> post) is really, I believe, the only one that return only different index!
>
>
> I don't think so. :-)
>
> Cheers,
>
> David

What do you mean? that there is another method or that it can not return
more than once the same index? ... I would be interested in knowning
another method!!!

IDL> a = (Round(Randomu(seed, 100) * 1000))
IDL> b=histogram(a)
IDL> print, where(b ge 2)
347 683 900 901 925

Jean

Subject: Re: Random selection
Posted by David Fanning on Sat, 11 Nov 2006 22:09:58 GMT
View Forum Message <> Reply to Message

Jean H. writes:

> What do you mean? that there is another method or that it can not return
> more than once the same index? ... I would be interested in knowning
> another method!!!
>
> IDL> a = (Round(Randomu(seed, 100) * 1000))
> IDL> b=histogram(a)
> IDL> print, where(b ge 2)
> 347 683 900 901 925

Oh, sorry. I guess I misunderstood what you meant.
I don't know if there are other methods. Ken's
certainly does give you unique integers. I guess
my only question is whether this method produces
a "random" selection. In other words, is a selection
that guarantees unique integers a "random" selection?

(I am having deja vu writing this, so it seems likely
we have discussed this in the past. I can't remember
what was decided.)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Random selection
Posted by Kenneth P. Bowman on Sun, 12 Nov 2006 04:09:45 GMT
View Forum Message <> Reply to Message

In article <MPG.1fbff8149c4b0217989db8@news.frii.com>,
David Fanning <news@dfanning.com> wrote:

> Jean H. writes:

>
>> What do you mean? that there is another method or that it can not return
>> more than once the same index? ... I would be interested in knowning
>> another method!!!
>>
>> IDL> a = (Round(Randomu(seed, 100) * 1000))

The problem with this approach is that it is actually rather likely
that you will generate two (or more) numbers that round to the
same 3-digit integer.

Sorting floats, on the other hand, involves no loss of precision.

If you really want to avoid duplicates, generate double-precision
random numbers

r = RANDOMU(seed, 1000, /DOUBLE)

Cheers, Ken

Subject: Re: Random selection
Posted by Julio[1] on Sun, 12 Nov 2006 12:57:45 GMT
View Forum Message <> Reply to Message

Thank you very much for all the comments. The problem was solved, I'm
using some tips that I found in the messages you sent.

Cheers,
Julio

greg michael escreveu:

> Try:
>
> result=array[*,randomu(seed,100)*1000]
>
> regards,
> Greg

Subject: Re: Random selection
Posted by greg michael on Sun, 12 Nov 2006 13:07:19 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman wrote:

> The problem with this approach is that it is actually rather likely
> that you will generate two (or more) numbers that round to the
> same 3-digit integer.
>
> Sorting floats, on the other hand, involves no loss of precision.
>
> If you really want to avoid duplicates, generate double-precision
> random numbers
>
> r = RANDOMU(seed, 1000, /DOUBLE)
>
> Cheers, Ken

I'm not sure doubles are going to help - you'll get the same problem at
3 digits. And if you use the sort method, it doesn't really matter if
you get two the same - they'll still map to unique indices.

many greetings,
Greg

Subject: Re: Random selection
Posted by JD Smith on Mon, 13 Nov 2006 18:46:08 GMT
View Forum Message <> Reply to Message

On Sun, 12 Nov 2006 05:07:19 -0800, greg michael wrote:

> I'm not sure doubles are going to help - you'll get the same problem at
> 3 digits. And if you use the sort method, it doesn't really matter if
> you get two the same - they'll still map to unique indices.

If you only want to pick 10 random elements from a list 100,000 long,
it's very inefficient to generate a set of 100,000 random numbers, sort
them all, and then take the first 10 indices. There are all sorts of
iterative higher-order algorithms for selection without replacement, but
they don't match to IDL well. One simple trick would be to start by
generating M random numbers, check for duplicates, and generate M-n more,
accumulating until you have enough.

```
M=10
len=100000L
inds=lonarr(n,/NOZERO)
n=M
while n gt 0 do begin
   inds[M-n]=long(randomu(sd,n)*len)
   inds=inds[sort(inds)]
   u=uniq(inds)
   n=M-n_elements(u)
```

```
    inds[0]=inds[u]
end
```

For this case, the speedup is immense, on average about 3500x faster. What about a case with more duplicates likely? How about len=100000, M=25000?

```
Sort All randoms:            0.13349121
Brute force replacement:     0.091892505
```

Still about 1.5x faster.

Obviously, if you wanted len-1 random indices, this won't scale, but in that case, you could just invert the problem, choose the random indices to be *discarded*, and use HISTOGRAM to generate the "real" list. Here's a general function which does this for you.

```
function random_indices, len, n_in
  swap=n_in gt len/2
  if swap then n=len-n_in else n=n_in
  inds=lonarr(n,/NOZERO)
  M=n
  while n gt 0 do begin
    inds[M-n]=long(randomu(sd,n)*len)
    inds=inds[sort(inds)]
    u=uniq(inds)
    n=M-n_elements(u)
    inds[0]=inds[u]
  endwhile

  if swap then inds=where(histogram(inds,MIN=0,MAX=len-1) eq 0)
  return,inds
end
```

It is outperformed by the simple sort method:

```
   r=randomu(sd,len)
   inds=(sort(r))[0:M-1]
```

only when M is close to len/2. For example, I found that selecting from length 100000 fewer than 30000 or more than 70000 elements favored RANDOM_INDICES. At worst case (M=len/2), it's roughly 3x slower. The RANDOM_INDICES method also returns the indices in sorted order (which you may or may not care about). You could obviously also make a hybrid approach which switches from one method to the other for

```
 abs(M-len/2) lt len/5
```

or so, but the tuning would be machine-specific.

JD