Subject: Re: "bootstrap" statistics
Posted by ejensen1 on Tue, 21 May 2002 21:39:24 GMT
View Forum Message <> Reply to Message

Wayne Landsman <landsman@mpb.gsfc.nasa.gov> wrote
>
> Interesting question. I had modified the above PERMUTE program to use a
> vector call to RANDOMN(/DOUBLE), (the /DOUBLE keyword has been available since
> V5.4). My feeling was that with ~5e15 distinct double precision numbers
> between 0 and 1, that probablity of RANDOMN returning two identical numbers was
> vanishingly small, in a typical call of say less than 10,000 numbers
>
> But I've never been comfortable enough with the modification to actually use
> it. I suppose I should add a check for any equal numbers in the
> RANDOMN(/DOUBLE) call, and then randomize those numbers.
>
> ;
> ; Select M numbers at random, repeating none.
>
> if N_elements(rseed) GT 0 then seed = rseed
> return, sort(randomn(seed, m,/Double))

This is the approach that I hit upon independently when I was doing
resampling statistics. Note that the possible repetition, unlikely
as it is, doesn't have much of an adverse effect on your resampling
since SORT returns distinct indices for tied values, e.g.

```
IDL> print, sort([3,1,3,3,3])
           1           0           2           3           4
```

So I do something like this:

```
; Return indices for a random subsample of size m out of a full
; population of size n without replacement:

 (sort(randomu(seed,n)))[0:m-1]
```

If there are any ties in the random numbers returned, the SORT call
will still return distinct indices, but it will be preserving a
small amount of information about the ordering of the parent
sample. If one really wanted to be paranoid about it, one could
add an extra randomization step in there, so you don't draw the first
m out of your final (mostly randomized) indices for the parent sample,
but you use that set of m indices that are between 0 and n-1
to tell you which of *another* set of randomized 0 to n-1
indices to pick:

```
(sort(randomu(seed,n)))[(sort(randomu(seed,n)))[0:m-1]]
```

Now as long as the repeats in one randomu call don't match up with the repeats in the other, I think you should be fine. In theory one would need an infinite regression of such randomizations, I suppose, to be completely free of this problem, but I think this should work pretty well for most applications. This should especially be true if you use the /Double keyword as Wayne Landsman suggested to minimize the chance of repetions, though I don't know how that effects speed relative to not using it.

Also, RANDOMN may be somewhat less likely to return repeated values than RANDOMU since the values aren't confined to 0-1, but on the other hand they are more strongly peaked around 0, so I'm not sure which one wins out. You could test it, which would also give you an idea of how often repeats happen (though I'm not sure off the top of my head the most efficient way to look for repeats in a big dataset).

Be sure to keep track of your SEED value if you have this in a subroutine, or you'll really have problems with non-randomness if you call RANDOMU repeatedly with an uninitialized SEED.

Hope this helps,

Eric
