Subject: Re: "bootstrap" statistics
Posted by Richard Younger on Thu, 30 May 2002 23:04:55 GMT
View Forum Message <> Reply to Message

Wayne Landsman wrote:
>
>
> Though the above method gives distinct values, they may not be completely random if
> the original RANDOMU() call returns duplicate values.

[...]

> However, the likelihood of RANDOMU returning duplicate values seems quite small for
> reasonable (<10,000?) sizes, and will be even smaller if the /DOUBLE keyword is used.
>
> Cheers , --Wayne Landsman

This piqued my interest a bit, as it seemed there must be a way to do it
in generally O(n) time rather than in the O(n lg n) that SORT requires.
I did a bit of looking in the definitive resource, and came up with (and
vectorized) the following.

It's algorithmically faster (assuming that WHERE runs in O(n)), but I
have no clue if it actually runs faster than the SORT method, due to a
larger number of O(n) calls and IDL's slightly funky nature, so YMMV.

On the plus side, it only calls RANDOMU M-n times instead of M, and it
doesn't have that duplication problem that the SORT method does, so test
it out if you need to squeeze out that last little bit of speed.

Best,
Rich

--
Richard Younger

```
;+
; NAME:
;       Kpick
; PURPOSE:
;       Randomly select n elements of a vector.
; USAGE:
;       NewIndex = Kpick( M, n, [SEED = ] )
; INPUT:
;       M = length of vector
;       n = number of elements to select
;
; OPTIONAL INPUT-OUTPUT:
```

```
;       SEED = random number seed to be passed to RANDOMU
;              Upon return, it will updated as a vector
;              containing a new seed
; OUTPUT:
;       Kpick returns n randomly shuffled integers between 0 and M-1
; EXAMPLE:
;       To select N elements of a vector V,
;
;       V = V[ Kpick(N_ELEMENTS(V), N) ]
;
; METHOD:
;       This routine is a vectorized form of the following literal
;       interpretation of Knuth's algorithm R.
;
;       I = LINDGEN(n)
;       FOR t=n, Set-1 DO BEGIN
;          M = RANDOMU(Seed, /long) MOD t
;          IF M LT n THEN BEGIN
;             I[M] = t
;          ENDIF
;       ENDFOR
;       RETURN, I
;
; REVISION HISTORY:
;       Written,  Richard Younger, MIT/LL 5/2002
;  Based on Algorithm R, from Knuth, D., The Art of
;              Computer Programming, Vol 2, Ch. 3.4.2
; RY 5/2002 Replaced random index generating line based on
;  MOD with slower but more statistically correct
;  statement based on floating point #s and FLOOR().
;  Feel free to use the faster MOD if you're nowhere near
;  2^31 ~ 2E9 elements.
;-

FUNCTION Kpick, Set, n, SEED=seed

 COMPILE_OPT idl2

 I = LINDGEN(n)

 ;generate a vector of uniform random #s between 0 and
       ;    [n,...,Set]
 ;M = RANDOMU(Seed, Set-n, /long) MOD (lindgen(Set-n)+n)
 M = FLOOR(RANDOMU(Seed, Set-n, /double)*(lindgen(Set-n)+n))

 t_pr = WHERE(M LT n)     ; t_pr + n = Knuth's t
 I[M[t_pr]] = t_pr+n
```

```
RETURN, I

END
```