Subject: "bootstrap" statistics
Posted by wmconnolley on Mon, 20 May 2002 19:41:44 GMT
View Forum Message <> Reply to Message

Hello group. I want to do what I think of as "bootstrap" statistics, viz
given a timeseries I take a random subsample (with, say, half the number
of elements), compute some statistic (say, then mean); then take another
random subsample; then again lots of times (say 1000 or 10000) and end
up with a distribution of the statistic concerned.

So: to do this I need a means to generate n/2 random (non-repeating)
indices from 0...n-1. At the moment I do this by:
once I have m indices I generate one more at
random; see if its in the list of m; if not, good; if it is, generate another
one. This is hideously inefficient and slow: there *must* be a better way.

I have found:

http://www.astro.washington.edu/deutsch-bin/getpro/library14 .html?PERMUTE

which is a somewhat better way, but still slow. Is there a no-loops version?

-W.

--
William M Connolley | wmc@bas.ac.uk | http://www.nerc-bas.ac.uk/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
I'm a .signature virus! copy me into your .signature file & help me spread!

Subject: Re: "bootstrap" statistics
Posted by Med Bennett on Sat, 25 May 2002 14:43:19 GMT
View Forum Message <> Reply to Message

wmc@bas.ac.uk wrote:

> Hello group. I want to do what I think of as "bootstrap" statistics, viz
> given a timeseries I take a random subsample (with, say, half the number
> of elements), compute some statistic (say, then mean); then take another
> random subsample; then again lots of times (say 1000 or 10000) and end
> up with a distribution of the statistic concerned.
>
> So: to do this I need a means to generate n/2 random (non-repeating)
> indices from 0...n-1. At the moment I do this by:
> once I have m indices I generate one more at
> random; see if its in the list of m; if not, good; if it is, generate another
> one. This is hideously inefficient and slow: there *must* be a better way.
>

> I have found:
>
> http://www.astro.washington.edu/deutsch-bin/getpro/library14 .html?PERMUTE
>
> which is a somewhat better way, but still slow. Is there a no-loops version?
>
> -W.
>
> --
> William M Connolley | wmc@bas.ac.uk | http://www.nerc-bas.ac.uk/icd/wmc/
> Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
> I'm a .signature virus! copy me into your .signature file & help me spread!

The way to do this without using the same value more than once is to use the sort command. First, generate m uniform random numbers, then find the sort indices of the random numbers. For example, if you want to randomly pick 50 out of 100 values:

```
IDL> junk = randomu(seed,100)
IDL> s=sort(junk)
IDL> print,s[0:49]
          1          96          16          19          69          97
85          91          59
         54          77           2          95          52          22
88           0          39
         44          70           8          63          50          82
41          43          42
         57          68          98          20          46          26
60          94          12
         35          72          51          14          71          64
78          29          89
         83          10          92          58          75
```

--

Remove all w's from my email address to reply!

ICBM Coordinates: 40.027666N, 105.289188W, 1670 m

Subject: Re: "bootstrap" statistics
Posted by wmconnolley on Sat, 25 May 2002 19:10:48 GMT
View Forum Message <> Reply to Message

Med Bennett <mwbennett@windra.com> wrote:
> The way to do this without using the same value more than once is to use the sort

> command. First, generate m uniform random numbers, then find the sort indices of
> the random numbers. For example, if you want to randomly pick 50 out of 100
> values:

```
> IDL> junk = randomu(seed,100)
> IDL> s=sort(junk)
> IDL> print,s[0:49]
```

Ah. Yes. Clever. (I have a feeling thats what one other post was trying to tell
me but wasn't so clear).

-W.

--
William M Connolley | wmc@bas.ac.uk | http://www.nerc-bas.ac.uk/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
I'm a .signature virus! copy me into your .signature file & help me spread!

Subject: Re: "bootstrap" statistics
Posted by Wayne Landsman on Mon, 27 May 2002 19:55:17 GMT

View Forum Message <> Reply to Message

wmc@bas.ac.uk wrote:

```
>  Med Bennett <mwbennett@windra.com> wrote:
>> The way to do this without using the same value more than once is to use the sort
>> command.  First, generate m uniform random numbers, then find the sort indices of
>> the random numbers.  For example, if you want to randomly pick 50 out of 100
>> values:
>
>> IDL> junk = randomu(seed,100)
>> IDL> s=sort(junk)
>> IDL> print,s[0:49]
>
>  Ah. Yes. Clever. (I have a feeling thats what one other post was trying to tell
>  me but wasn't so clear).
```

Though the above method gives distinct values, they may not be completely random if
the original RANDOMU() call returns duplicate values.   The behaviour of the
SORT() command when encountering duplicate values differs on different machine
architectures, but on many machines the original array order is maintained (e.g.
SORT([4,3,4,4,2]) returns [4,1,0,2,3].   This introduces a bias, where lower
indicies are more likely to occur at the beginning of the permutation vector, and
thus may bias a result that depends on completely random permuations.

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

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
```