Subject: Re: Generating a grid in the 3D,4D,5D...N space - Advice/Combinatory/Matrices
Posted by Dick Jackson on Tue, 14 Nov 2017 18:13:58 GMT
View Forum Message <> Reply to Message

Hi Clement,

I found a few efficiencies to improve Markus' method a little, and I think I have a working method to allow unequal numbers of rows and columns—it could be improved, but I have to leave this for the moment.

Report from running the tests:

Markus' method
% Time elapsed: 0.15960312 seconds.
n = 7
Memory used (MB): 12.5667

Dick's method 1
% Time elapsed: 0.19358516 seconds.
n = 7
Memory used (MB): 5.49789
Arrays match!

Dick's method 2
% Time elapsed: 0.34163213 seconds.
nc = 7
nr = 7
Memory used (MB): 12.5669
Arrays match!

Here is the code:

```
PRO MultiDimCombos

;  https://groups.google.com/forum/#!topic/comp.lang.idl-pvwave /FV6s1s19BJc

n = 7

; Markus

startMem = Memory(/CURRENT)
Tic

array=bindgen(n,long(n)^n)
for k=0,n-1 do $
  array[k,*]=rebin(byte((n*lindgen(long(n)^(k+1))+k) mod (n^2) ), $
                long(n)^n, /sample)
```

```
Print
Print, 'Markus'' method'
Toc
highMem = Memory(/HIGHWATER)
Print, 'n = '+StrTrim(n, 2)
Print, 'Memory used (MB): ', (highMem-startMem)/1024./1024
arrayM = array

IF n LE 3 THEN Print, array

; Dick 1

startMem = Memory(/CURRENT)
Tic

array=bytarr(n,long(n)^n, /NOZERO) ; Changed from bindgen to bytarr(/NOZERO)
for k=0,n-1 do $ ; Removed '*' from destination subscripts
  array[k,0]=Reform(rebin(byte((n*lindgen(long(n)^(k+1))+k) mod (n^2) ), $
                          long(n)^n, /sample), $
                    [1, long(n)^n], /OVERWRITE)

Print
Print, 'Dick''s method 1'
Toc
highMem = Memory(/HIGHWATER)
Print, 'n = '+StrTrim(n, 2)
Print, 'Memory used (MB): ', (highMem-startMem)/1024./1024
arrayD1 = array

IF n LE 3 THEN Print, array

Print, 'Arrays'+([' do not', ''])[Array_Equal(arrayD1, arrayM)]+' match!'

; Dick 2 -- method for unequal numbers of columns and rows

startMem = Memory(/CURRENT)
Tic

; To compare with Markus and Dick1:
nc = n
nr = n

; To test unequal nc and nr:
;nc = 4
;nr = 3

a = bindgen(nc, nr)   ; bindgen is OK to nr = 8
```

```
i = lindgen([1, Long(nr)^nc]) ; indgen is OK to nr = 8
array=bytarr(nc,long(nr)^nc, /NOZERO)
for k=0,nc-1 do $
  array[k,0]=a[k,i/(Long(nr)^(nc-1-k)) MOD nr]

Print
Print, 'Dick''s method 2'
Toc
highMem = Memory(/HIGHWATER)
Print, 'nc = '+StrTrim(nc, 2)
Print, 'nr = '+StrTrim(nr, 2)
Print, 'Memory used (MB): ', (highMem-startMem)/1024./1024
arrayD2 = array

IF nr * nc LE 12 THEN Print, array

IF nc EQ n and nr EQ n THEN $
  Print, 'Arrays'+([' do not', '']))[Array_Equal(arrayD2, arrayM)]+' match!'

END
```

Hope this helps!

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada --- http://www.d-jackson.com

On Tuesday, 14 November 2017 05:16:47 UTC-8, clement...@obspm.fr wrote:
> Hello again,
>
> Thanks both of you for your replies.
>
> @Mike: I had looked into this before (I think Jeremy Bailin has published a code similar to yours called combigen.pro), but I then meet difficulties in selecting part of the generated combinations.
>
> @Markus: I say, your code is sleek and nifty. I like your solution.
>
> In the meantime, I had given this problem some more thoughts and I had come up with another slow ugly one that doesn't work for all cases:
>
> function gen_indices_comb, m, n
> ;d I/O:
> ;d m -> long integer corresponds to the number of row in original table
> ;d n -> long integer corresponds to the number of columns in original table
> ;d

```
>  ;d vals -> long array listing the vectors of indices to extract the
>  ;d         different possible combinations from the values of the original
>  ;d         table
>  ;d
>  ;d NOTES: SLOW CODE, a mitigation of the values of m and n is REQUIRED
>  ;d        Cases, where m & n are greater than 9, are not to considered
>  ;d        with this code
>
>   nmax = m^n
>
>  ;c Assemble command generating vector of indices
>   cmd = 'tmp = ['
>   for ijk=(m-1L),1L,-1L do $
>     cmd += ' (lmn/n^'+string(ijk,format='(I03)')+') mod n,'
>   cmd += 'lmn mod n ]'
>
>  ;c initialiase memory
>   ini = indgen(m,n)
>   tmp = lonarr(m)
>   val = lonarr(m, m^n)
>
>  ;c execute command for each type of combination
>   for lmn=0L,(n^m-1L) do begin
>     void = execute(cmd)
>     if void ne 1 then message, ' > Error generating indices.'
>     val[*,ijk] = ini+tmp*m
>   endfor
>
>   return, val
>  end
>
>  Afn I'm considering this post solved, I'll update it with a definitive version of my solution.
>
>  Again thanks your replies,
>  /C
```