Subject: Re: m choose n
Posted by Jeremy Bailin on Thu, 13 Aug 2009 03:21:43 GMT
View Forum Message <> Reply to Message

On Aug 10, 9:43 am, Paolo <pgri...@gmail.com> wrote:
> On Aug 8, 9:57 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
>
>
>
>
>
>> On Jul 29, 9:38 am, Paolo <pgri...@gmail.com> wrote:
>
>>> On Jul 28, 7:09 pm, Rob <rob.webina...@gmail.com> wrote:
>
>>>> Has anyone implemented the combinatorial function which the "n choose
>>>> k" combinations of an input vector, like Matlab's nchoosek? I'm not
>>>> talking about just the binomial coefficient n!/(m!*(n-m)!). I'm
>>>> interested in getting the "n choose k" combinations. Matlab's
>>>> function:
>
>>>> http://www.mathworks.com/access/helpdesk/help/techdoc/index. html?/acc...
>
>>>> Example:
>>>> octave-3.0.5:2> nchoosek([1,2,3,4],2)
>>>> ans =
>
```
>>>>    1   2
>>>>    1   3
>>>>    1   4
>>>>    2   3
>>>>    2   4
>>>>    3   4
```
>
>>>> If not, I will just codify Matlab/Octave's nchoosek() and submit to
>>>> ITT Vis or something like that.
>
>>>> R
>
>>> Yes, I posted this function to the newsgroup a few years ago.
>
>>> http://tinyurl.com/nra4d8
>
>>> I report it below.
>
>>> To reproduce your result:
>>> a=[1,2,3,4]
>>> combind=pgcomb(4,2)

```
>>>  print,a[combind]
>>>  or
>>>  print,pgcomb(4,2)+1 if you are lazy :)
>
>>>  It's a nice example of a routine that would be
>>>  somewhat harder to write without a BREAK statement :)
>
>>>  Ciao,
>>>  Paolo
>
>>>  FUNCTION pgcomb,n,j
>>>  ;;number of combinations of j elements chosen from n
>>>  nelres=long(factorial(n)/(factorial(j)*factorial(n-j)))
>
>>>  res=intarr(j,nelres);array for the result
>>>  res[*,0]=indgen(j);initialize first combination
>
>>>  FOR i=1,nelres-1 DO BEGIN;go over all combinations
>>>     res[*,i]=res[*,i-1];initialize with previous value
>
>>>     FOR k=1,j DO BEGIN;scan numbers from right to left
>
>>>      IF res[j-k,i] LT n-k THEN BEGIN;check if number can be increased
>
>>>         res[j-k,i]=res[j-k,i-1]+1;do so
>
>>>         ;if number has been increased, set all numbers to its right
>>>         ;as low as possible
>>>         IF k GT 1 THEN res[j-k+1:j-1,i]=indgen(k-1)+res[j-k,i]+1
>
>>>         BREAK;we can skip to the next combination
>
>>>        ENDIF
>
>>>     ENDFOR
>
>>>  ENDFOR
>
>>>  RETURN,res
>
>>>  END
>
>>  Here's a vectorized version... probably less efficient in most regions
>>  of parameter space, but might be better if k isn't too large and the
>>  number of combinations is large:
>
>>  IDL> a = [1,2,3,4]
>>  IDL> n = n_elements(a)
>>  IDL> k = 2L
>>  IDL> q = array_indices(replicate(n,k),lindgen(n^k),/dimen)
>>  IDL> print, a[q[*,where(min(q[1:k-1,*]-q[0:k-2,*],dimen=1) gt 0)]]
>>        1       2
>>        1       3
>>        2       3
>>        1       4
>>        2       4
>>        3       4
>>  IDL> k = 3L
>>  IDL> q = array_indices(replicate(n,k),lindgen(n^k),/dimen)
>>  IDL> print, a[q[*,where(min(q[1:k-1,*]-q[0:k-2,*],dimen=1) gt 0)]]
>>        1       2       3
>>        1       2       4
>>        1       3       4
>>        2       3       4
>
>>  -Jeremy.
>
>  Isn't it amazing what IDL can do if you throw memory at the problem?
>
>  Now that would be so cool if we didn't have to create that k by n^k
>  array :)
>
>  Ciao,
>  Paolo
```

Heheh... yeah, I know. :-)= Still, it might be more efficient than
the for loop for small k and large n.

Actually, you can do a lot better for large k by using the
complementarity of "n choose k" and "n choose (n-k)"... if n-k is
smaller, then first find the combinations for n choose (n-k), and then
use some histogram magic to find the complement of each set. That way
you don't need to generate enormous arrays to do things like 10 choose
9. ;-) Here's an implementation:

```
function nchoosek, n, k

nl = long(n)
kl = long(k)
if kl gt nl/2 then begin
  kl=nl-k
  kcomplement=1
endif else kcomplement=0

q = array_indices(replicate(nl,kl),lindgen(nl^kl),/dimen)
```

```
if kl ne 1 then combi = q[*,where(min(q[1:kl-1,*]-q[0:kl-2,*],dimen=1)
gt 0)] $
  else combi = reform(q, 1, nl^kl)

; if k > n/2, find the complementary set using a pseudo-2D histogram
if kcomplement then begin
  s = size(combi,/dimen)
  ncombi = s[1]
  index2d = combi + rebin(reform(lindgen(ncombi)*nl,
1,ncombi),kl,ncombi)
  return, reform(where(histogram(index2d, min=0, max=nl*ncombi-1) eq
0) mod nl, k, ncombi)
endif else return, combi

end
```

-Jeremy.