
Subject: Re: m choose n

Posted by [Jeremy Bailin](#) on Sun, 09 Aug 2009 01:57:59 GMT

[View Forum Message](#) <> [Reply to Message](#)

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.
