
Subject: Cartesian product of a list of lists

Posted by [Fabzi](#) on Sun, 04 May 2014 15:48:52 GMT

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

Folks,

to test all possible parameter combinations for a model run I needed following algorithm (using lists!):

<http://stackoverflow.com/questions/2419370/how-can-i-compute-a-cartesian-product-iteratively>

While I was successful in implementing the IDL version of the *iterative* algorithm (first answer to the above mentioned post), I miserably failed to implement the *recursive* one. It seems easy but it's Sunday and I can't figure it out.

Anyway, if someone is interested here's the iterative version. If anyone wants to try the recursive version I'd be interested to see how it works ;-)

```
;+
;:Description:
;  Computes the Cartesian product of a given number of vectors stored
;  in a list. The list can be a list of arrays or a list of lists.
;
;  Iterative algorithm obtained here:
;  http://stackoverflow.com/questions/2419370/
;  how-can-i-compute-a-cartesian-product-iteratively
;
;:Params:
;  inList: in, required
;          a list of arrays to combine
;
;:Returns:
;  a list of arrays with all possible combinations
;
;:Examples:
;  Try::
;  IDL> I = LIST([1,2,3], [4,5],[6,7])
;  IDL> print, w_cartesianProduct(I)
;
;  IDL should print::
;
;      1      4      6
;      1      4      7
;      1      5      6
;      1      5      7
;      2      4      6
;      2      4      7
```

```

;      2      5      6
;      2      5      7
;      3      4      6
;      3      4      7
;      3      5      6
;      3      5      7
;
;
; :Author: Fabien Maussion 2014
;      Last modification: FaM, May 4, 2014
;
;
;-
function w_cartesianProduct, inList

```

```

    result = list()

    nl = inlist.count()
    combis = lonarr(nl)
    indexes = lonarr(nl)
    while 1 do begin
        for i=0, nl-1 do combis[i] = (inlist[i])[indexes[i]]
        result->add, combis
        j = nl - 1
        while 1 do begin
            indexes[j] += 1
            if indexes[j] lt n_elements(inlist[j]) then break
            indexes[j] = 0
            j -= 1
            if j lt 0 then return, result
        endwhile
    endwhile
end

```

end

Subject: Re: Cartesian product of a list of lists
 Posted by [Fabzi](#) on Sun, 04 May 2014 16:53:10 GMT
[View Forum Message](#) <> [Reply to Message](#)

Sorry for the double post. Here is a version which is more flexible as it returns a list of lists:

```

;+
; :Description:
;   Computes the Cartesian product of a given number of vectors stored
;   in a list. The list can be a list of arrays or a list of lists.
;
;   Iterative algorithm obtained here:
;   http://stackoverflow.com/questions/2419370/how-can-i-compute -

```

```

;      a-cartesian-product-iteratively
;
;
;:Params:
;  inList: in, required
;          a list of arrays/lists to combine
;
;:Returns:
;  a list of lists with all possible combinations
;
;:Examples:
;  Try::
;  IDL> l = LIST([1,2,3], [4,5],[6,7])
;  IDL> foreach el, w_cartesianProduct(l) do print, el->toArray(TYPE=7)
;
;      1      4      6
;      1      4      7
;      1      5      6
;      1      5      7
;      2      4      6
;      2      4      7
;      2      5      6
;      2      5      7
;      3      4      6
;      3      4      7
;      3      5      6
;      3      5      7
;
;  IDL> l = LIST([1,2,3], ['T4','T5'],[6,7])
;  IDL> foreach el, w_cartesianProduct(l) do print, el->toArray(TYPE=7)
;
;      1 T4      6
;      1 T4      7
;      1 T5      6
;      1 T5      7
;      2 T4      6
;      2 T4      7
;      2 T5      6
;      2 T5      7
;      3 T4      6
;      3 T4      7
;      3 T5      6
;      3 T5      7
;
;:Author: Fabien Maussion 2014
;      Last modification: FaM, May 4, 2014
;
;-
function w_cartesianProduct, inList

    result = list()

```

```
nl = inlist.count()
indexes = lonarr(nl)
while 1 do begin
  li = list()
  for i=0, nl-1 do li->add, (inlist[i])[indexes[i]]
  result->add, li
  j = nl - 1
  while 1 do begin
    indexes[j] += 1
    if indexes[j] lt n_elements(inlist[j]) then break
    indexes[j] = 0
    j -= 1
    if j lt 0 then return, result
  endwhile
endwhile
end
```
