Subject: Re: REPLICATE with arrays
Posted by Craig Markwardt on Fri, 11 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) writes:
> Vince Hradil (hradilv@yahoo.com) writes:
>
>> I often have the need to replicate an array, but IDL's replicate only
>> works with scalars. Does anyone have any tips on the most efficient,
>> simplest, clearest (you choose) way to do this?
>
> I am such a sucker for these kinds of articles. :-(
> ...

I'll add my implementation to the competition. Here is CMREPLICATE which takes either a scalar or array. It uses the REFORM/REBIN magic that has already been discussed, for numeric types. For other types I did have to revert to MAKE_ARRAY/copy technique, but I've sped up the copy vs a simple for loop. Obfuscatory I am sure!

It's also available at my web page (and if you are a regular check the NEWS too, since I've made a few other updates):

http://cow.physics.wisc.edu/~craigm/idl/idl.html

Craig

```
;+
; NAME:
;   CMREPLICATE
;
; AUTHOR:
;   Craig B. Markwardt, NASA/GSFC Code 662, Greenbelt, MD 20770
;   craigm@lheamail.gsfc.nasa.gov
;
; PURPOSE:
;   Replicates an array or scalar into a larger array, as REPLICATE does.
;
; CALLING SEQUENCE:
;   ARRAY = CMREPLICATE(VALUE, DIMS)
;
; DESCRIPTION:
;
;   The CMREPLICATE function constructs an array, which is filled with
;   the specified VALUE template.  CMREPLICATE is very similar to the
;   built-in IDL function REPLICATE.  However there are two
;   differences:
;
```

```
;     * the VALUE can be either scalar or an ARRAY.
;
;     * the dimensions are specified as a single vector rather than
;       individual function arguments.
;
;   For example, if VALUE is a 2x2 array, and DIMS is [3,4], then the
;   resulting array will be 2x2x3x4.
;
; INPUTS:
;
;   VALUE - a scalar or array template of any time, to be replicated.
;           NOTE: These two calls do not produce the same result:
;                 ARRAY = REPLICATE( 1,  DIMS)
;                 ARRAY = REPLICATE([1], DIMS)
;           In the first case the output dimensions will be DIMS and
;           in the second case the output dimensions will be 1xDIMS
;           (except for structures).  That is, a vector of length 1 is
;           considered to be different from a scalar.
;
;   DIMS - Dimensions of output array (which are combined with the
;          dimensions of the input VALUE template).  If DIMS is not
;          specified then VALUE is returned unchanged.
;
; RETURNS:
;   The resulting replicated array.
;
; EXAMPLE:
;   x = [0,1,2]
;   help, cmreplicate(x, [2,2])
;     <Expression>    INT       = Array[3, 2, 2]
;   Explanation: The 3-vector x is replicated 2x2 times.
;
;   x = 5L
;   help, cmreplicate(x, [2,2])
;     <Expression>    LONG      = Array[2, 2]
;   Explanation: The scalar x is replicated 2x2 times.
;
; SEE ALSO:
;
;   REPLICATE
;
; MODIFICATION HISTORY:
;   Written, CM, 11 Feb 2000
;
;-
; Copyright (C) 2000, Craig Markwardt
; This software is provided as is without any warranty whatsoever.
; Permission to use, copy, modify, and distribute modified or
```

```
; unmodified copies is granted, provided this copyright and disclaimer
; are included unchanged.
;-
function cmreplicate, array, dims

  if n_params() EQ 0 then begin
     message, 'RARRAY = CMREPLICATE(ARRAY, DIMS)', /info
     return, 0L
  endif

  if n_elements(dims) EQ 0 then return, array
  if n_elements(array) EQ 0 then $
    message, 'ERROR: ARRAY must have at least one element'

  ;; Construct new dimensions, being careful about scalars
  sz = size(array)
  type = sz(sz(0)+1)
  if sz(0) EQ 0 then return, make_array(value=array, dimension=dims)
  onedims = [sz(1:sz(0)), dims*0+1] ;; For REFORM, to extend # of dims.
  newdims = [sz(1:sz(0)), dims    ] ;; For REBIN, to enlarge # of dims.
  nnewdims = n_elements(newdims)

  if nnewdims GT 8 then $
    message, 'ERROR: resulting array would have too many dimensions.'

  if type NE 7 AND type NE 8 AND type NE 10 AND type NE 11 then begin
     ;; Handle numeric types

     ;; Argghh!  Why doesn't REBIN take an *array* of dimensions!
     ;; *Instead* we need to run EXECUTE(), with a string like this:
     ;; rebin(array1, newdims(0), newdims(1), ...)
     ;; That's what the following format string does.
     fmt = '('+strtrim(nnewdims,2)+'("newdims(",I0,")",:,","))'
     arglist = string(lindgen(nnewdims), format=fmt)
     cmd = 'return, rebin(reform([array], onedims),'+arglist+')'
     dummy = execute(cmd)

     ;; If execution reaches here then an error occurred.
     message, 'ERROR: array could not be resized'
     return, 0L

  endif else begin
     ;; Handle strings, structures, pointers, and objects separately

     ;; Handle structures, which are never scalars
     if type EQ 8 AND sz(0) EQ 1 AND n_elements(array) EQ 1 then $
       return, make_array(value=array, dimension=dims)
```

```
    nold = n_elements(array)
    nadd = 1L
    for i = 0L, n_elements(dims)-1 do nadd = nadd * long(dims(i))
    array1 = make_array(type=sz(sz(0)+1), dimension=[nold,nadd])
    array2 = reform([array], n_elements(array))

    ;; Efficient copying, done by powers of two
    array1(0,0) = array2
    stride = 1L   ;; stride increase by a factor of two in each iteration
    i = 1L & nleft = nadd - 1
    while nleft GT stride do begin
       array1(0,i) = array1(*,0:stride-1)  ;; Note sneaky IDL optimization
       i = i + stride & nleft = nleft - stride
       stride = stride * 2
    endwhile
    if nleft GT 0 then array1(0,i) = array1(*,0:nleft-1)

    return, reform(array1, newdims, /overwrite)
 endelse

end
```

Subject: Re: REPLICATE with arrays
Posted by Liam E. Gumley on Fri, 11 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning <davidf@dfanning.com> wrote in message
news:MPG.130e00a4f5f14484989a29@news.frii.com...
> William Thompson (thompson@orpheus.nascom.nasa.gov) writes:
>
>> Stein Vidar Haugan once proposed the following strategy which appears to
be
>> highly efficient:
>>
>> x2 = rebin(reform(x,3,3,1), 3,3,2)
>
> And Liam Gumley (Liam.Gumley@ssec.wisc.edu) writes:
>
>> IDL> arr = rebin(reform(arr, 3, 3, 1), 3, 3, 2, /sample)
>
> I don't think the IDL newsgroup is the proper place to
> practice the occult sciences. I wish you guys would cut
> it out. :-(

I'll admit that two reshape operations in one line is a bad idea. But if
it's re-written as

arr = reform(arr, 3, 3, 1) ; add an extra dimension
arr = rebin(arr, 3, 3, 2, /sample) ; expand along the extra dimension

then I think it's about as clear as you can get (in the IDL world anyway).

Cheers,
Liam.

Subject: Re: REPLICATE with arrays
Posted by davidf on Fri, 11 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

William Thompson (thompson@orpheus.nascom.nasa.gov) writes:

> Stein Vidar Haugan once proposed the following strategy which appears to be
> highly efficient:
>
> x2 = rebin(reform(x,3,3,1), 3,3,2)

And Liam Gumley (Liam.Gumley@ssec.wisc.edu) writes:

> IDL> arr = rebin(reform(arr, 3, 3, 1), 3, 3, 2, /sample)

I don't think the IDL newsgroup is the proper place to
practice the occult sciences. I wish you guys would cut
it out. :-(

Cheers,

David

P.S. What the heck has happened to Stein Vidar!? He didn't
finally get his Ph.D. and move on to a real job, did he?

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: REPLICATE with arrays
Posted by thompson on Fri, 11 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Vince Hradil <hradilv@yahoo.com> writes:

> I often have the need to replicate an array, but IDL's replicate only
> works with scalars. Does anyone have any tips on the most efficient,
> simplest, clearest (you choose) way to do this?

> e.g.

> I have:
> help, x
>   INT   = Array[3, 3]
> print, x
>   2   4   10
>   3   7   5
>   3   9   2

> and would like to do:
> x2 = replicate(x,2)
> help, x2
>   INT   = Array[3, 3, 2]
> print, x2
>   2   4   10
>   3   7   5
>   3   9   2

>   2   4   10
>   3   7   5
>   3   9   2

> I've figured out some trick for 1 and 2 dimensional arrays, but I'm
> looking for a more general strategy to use on higher dim arrays.

Stein Vidar Haugan once proposed the following strategy which appears to be highly efficient:

```
x2 = rebin( reform(x,3,3,1), 3,3,2)
```

If one always wants the replication dimension to be the last dimension, then the following should be a good way of generalizing it:

```
sz = size(x)
dim = sz[1:sz[0]]
x2 = reform(x,[dim,1])
case n_elements(dim) of
   1: x2=rebin(x2,dim(0),n)
   2: x2=rebin(x2,dim(0),dim(1),n)
   3: x2=rebin(x2,dim(0),dim(1),dim(2),n)
   4: x2=rebin(x2,dim(0),dim(1),dim(2),dim(3),n)
```

```
    5: x2=rebin(x2,dim(0),dim(1),dim(2),dim(3),dim(4),n)
    6: x2=rebin(x2,dim(0),dim(1),dim(2),dim(3),dim(4),dim(5),n)
    7:  x2=rebin(x2,dim(0),dim(1),dim(2),dim(3),dim(4),dim(5),dim(6) ,n)
endcase
```

William Thompson

Subject: Re: REPLICATE with arrays
Posted by Liam E. Gumley on Fri, 11 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Vince Hradil <hradilv@yahoo.com> wrote in message
news:38A43130.D0ACB994@yahoo.com...

```
>  I often have the need to replicate an array, but IDL's replicate only
>  works with scalars.  Does anyone have any tips on the most efficient,
>  simplest, clearest (you choose) way to do this?
>
>  e.g.
>
>  I have:
>  help, x
>     INT    = Array[3, 3]
>  print, x
>     2    4    10
>     3    7    5
>     3    9    2
>
>  and would like to do:
>  x2 = replicate(x,2)
>  help, x2
>     INT    = Array[3, 3, 2]
>  print, x2
>     2    4    10
>     3    7    5
>     3    9    2
>
>     2    4    10
>     3    7    5
>     3    9    2
>
>  I've figured out some trick for 1 and 2 dimensional arrays, but I'm
>  looking for a more general strategy to use on higher dim arrays.
```

How about this:

```
IDL> arr = indgen(3, 3)
IDL> print, arr
```

```
      0      1      2
      3      4      5
      6      7      8
IDL> arr = rebin(reform(arr, 3, 3, 1), 3, 3, 2, /sample)
IDL> print, arr
      0      1      2
      3      4      5
      6      7      8

      0      1      2
      3      4      5
      6      7      8
```

REFORM adds an extra dimension, and REBIN expands the array along the extra dimension. This will work just fine for higher dimensions.

Cheers,
Liam.
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: REPLICATE with arrays
Posted by davidf on Fri, 11 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Vince Hradil (hradilv@yahoo.com) writes:

> I often have the need to replicate an array, but IDL's replicate only
> works with scalars. Does anyone have any tips on the most efficient,
> simplest, clearest (you choose) way to do this?

I am such a sucker for these kinds of articles. :-(

I don't know about "efficient, simplest, clearest", but
here is something that works for the three cases (1D, 2D,
and 3D arrays) I tried it on. I leave it to your imagination
to figure out the rest of the cases. It seemed like details
to me. :-)

Cheers,

David

```
 ************************************************************ ************
FUNCTION Replicate_Array, array, number

   ; Obtain information about the array.
```

```
ndims = Size(array, /N_Dimensions)
dims = Size(array, /Dimensions)
type = Size(array, /TName)

   ; Initialize some variables.

snumber = StrTrim(number,2)
newArray = 0

   ; Create the new array and populate it with the old array.

CASE ndims OF
  1: BEGIN
     command = 'newArray = Make_Array(' + StrTrim(dims[0],2) + ',' $
        + snumber + ',' + type + '=1)'
     ok = Execute(command)
     FOR j=0,number-1 DO newArray[*,j] = array
     END
  2: BEGIN
     command = 'newArray = Make_Array(' + StrTrim(dims[0],2) + ',' + $
        StrTrim(dims[1],2) + ',' + snumber + ',' + type + '=1)'
     ok = Execute(command)
     FOR j=0,number-1 DO newArray[*,*,j] = array
     END
  3: BEGIN
     command = 'newArray = Make_Array(' + StrTrim(dims[0],2) + ',' + $
        StrTrim(dims[1],2) + ',' + StrTrim(dims[2],2) + ','  + snumber $
        + ',' + type + '=1)'
     ok = Execute(command)
     FOR j=0,number-1 DO newArray[*,*,*,j] = array
     END
  ELSE: Print, 'Sorry, figure this out yourself. :-)'
ENDCASE

RETURN, newArray
END
 ************************************************************** ***********
```

David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155