Subject: Concatenating arrays across chosen dimension
Posted by Dick Jackson on Tue, 25 Jun 2002 22:05:01 GMT
View Forum Message <> Reply to Message

"Randall Skelton" <rhskelto@atm.ox.ac.uk> wrote in message
news:Pine.LNX.4.33.0206251749570.28170-100000@mulligan.atm.o x.ac.uk...

> Ok... I have to ask. Is there actually a nice, clean way to concatenate
> multidimensional arrays in IDL?
>
> a = make_array(2,2,2,2)
> b = make_array(2,2,2,5)
>
> data1 = [[[[a]]] , [[[b]]]]
>
> Obviously the above fails, but what is the solution? Surely some
> combination of rebin/reform...

Well, I have to say I don't know *why* that one fails, since this works
fine:

```
IDL> a = make_array(2,2,2)
IDL> b = make_array(2,2,5)
IDL> help, [ [[a]], [[b]] ]
<Expression>    FLOAT     = Array[2, 2, 7]
```

... and we're a long way from the 8-dimension limit on arrays.

In any case, to concatenate on the *last* dimension in this case:

```
c = Reform([ a[*], b[*] ], [2,2,2,7])
```

To do this in general takes a little more hacking, dimension juggling in
particular. I couldn't resist the challenge, so I whipped up the attached
Concat.pro.

Examples:

```
IDL> Help, Concat(LIndGen(2,3,4), -LIndGen(5,3,4), Dim=0)
<Expression>    LONG      = Array[7, 3, 4]
IDL> Help, Concat(LIndGen(2,3,4), -LIndGen(2,5,4), Dim=1)
<Expression>    LONG      = Array[2, 8, 4]
IDL> Help, Concat(LIndGen(2,3,4), -LIndGen(2,3,5)) ; assumes last dimension
<Expression>    LONG      = Array[2, 3, 9]
```

Code is copied here for convenience. Any comments or corrections are
welcome!

```
=====

FUNCTION Concat, $                  ; Return concatenation of two arrays
   a, $                             ; First array
   b, $                             ; Second array
   Dimension=dim                        ; Dimension along which to
concatenate
                                    ; (counting from 0, defaults to
*last*
                                    ; dimension of arrays)
;;    Examples:
;;    IDL> Help, Concat(LIndGen(2,3,4), -LIndGen(5,3,4), Dim=0)
;;    <Expression>    LONG      = Array[7, 3, 4]
;;    IDL> Help, Concat(LIndGen(2,3,4), -LIndGen(2,5,4), Dim=1)
;;    <Expression>    LONG      = Array[2, 8, 4]
;;    IDL> Help, Concat(LIndGen(2,3,4), -LIndGen(2,3,5))
;;    <Expression>    LONG      = Array[2, 3, 9]
;;    (use Print with these to see actual results)

;;    Assuming here that a and b are of same dimensions except perhaps for
;;    dimension 'dim', which is assumed to be within range.
;;    Testing and error handling for this is left as an exercise for the
;;    reader. :-)

nDims = Size(a, /N_Dimensions)
IF N_Elements(dim) EQ 0 THEN dim = nDims-1

aDims = Size(a, /Dimensions)
bDims = Size(b, /Dimensions)

;;    Figure out desired dimensions of result

resultDims = aDims
resultDims[dim] = aDims[dim]+bDims[dim]

;;    Make a vector of dimension indices with concatenation dimension *last*

transposeDimOrder = [Where(IndGen(nDims) NE dim), dim]

;;    Juggle dimensions by transposing a and b to put desired concatenation
;;    dimension last, then take all elements together into one vector

joinedVector = [(Transpose(a, transposeDimOrder))[*], $
            (Transpose(b, transposeDimOrder))[*]]

;;    Reform the vector to an array of the right size with juggled
dimensions
```

juggledResult = Reform(Temporary(joinedVector),
resultDims[transposeDimOrder])

;; Un-juggle the dimensions to give final result

Return, Transpose(Temporary(juggledResult), Sort(transposeDimOrder))

END

=====

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

Subject: Re: Concatenating arrays
Posted by Wonko[3] on Sun, 04 Sep 2005 19:06:00 GMT
View Forum Message <> Reply to Message

RPF@lol.com (RPF) wrote:

> If I have a = [2 3] and i wish to repeat these two elements for say 1000
> times so my final array is 2000 elements long is there a quick way?

Have a look at REPLICATE().

> Also is there an easy way to make an upper triangular matrix i.e.
> make all entries below the diagonal zero ?

I think CHOLDC does this.

  Alex
--
Alex Schuster Wonko@wonkology.org
  alex@pet.mpin-koeln.mpg.de

Subject: Re: Concatenating arrays
Posted by David Fanning on Sun, 04 Sep 2005 19:11:48 GMT
View Forum Message <> Reply to Message

RPF writes:

> me...
> If I have a = [2 3] and i wish to repeat these two elements for say 1000
> times so my final array is 2000 elements long is there a quick way?

You want to read the Dimensional Juggling Tutorial:

  http://www.dfanning.com/documents/tips.html#Tutorials

Try this:

```
a = [2, 3]
b = Reform(Rebin(a, 2, 1000), 2000)
```

> Also is there an easy way to make an upper triangular matrix i.e.
> make all entries below the diagonal zero ?

Probably. There always is. But the exactly method eludes me at
the moment. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Concatenating arrays
Posted by K. Bowman on Mon, 05 Sep 2005 14:08:51 GMT
View Forum Message <> Reply to Message

In article <431af10e@quokka.wn.com.au>, "RPF" <RPF@lol.com> wrote:

> Also is there an easy way to
> make an upper triangular matrix i.e. make all entries below the diagonal
> zero ?

```
IDL> n = 5
IDL> i = REBIN(LINDGEN(n), n, n)
IDL> j = REBIN(TRANSPOSE(LINDGEN(n)), n, n)
IDL> print, i
       0       1       2       3       4
       0       1       2       3       4
       0       1       2       3       4
       0       1       2       3       4
```

```
       0       1       2       3       4
IDL> print, j
       0       0       0       0       0
       1       1       1       1       1
       2       2       2       2       2
       3       3       3       3       3
       4       4       4       4       4
IDL> a = (i GE j)
IDL> print, a
  1   1   1   1   1
  0   1   1   1   1
  0   0   1   1   1
  0   0   0   1   1
  0   0   0   0   1
```

Cheers, Ken Bowman
