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