Subject: Array Concatenation Tutorial
Posted by John-David T. Smith on Tue, 01 May 2001 20:42:15 GMT
View Forum Message <> Reply to Message

This the second in my series of tutorials designed to relieve me from having to keep all of this information in my head. You might have read the former: "Dimensional Juggling Tutorial", which you can find at:

http://groups.google.com/groups?as_umsgid=3AC7B345.1648F0A9% 40astro.cornell.edu

Have you seen IDL statements which look like:

```
IDL> a=[[[b]],[[c]]]
```

Do you wonder what all those brackets are doing there? Is it some form of code feng shui, achieving harmonious spiritual balance through nested punctuation?

Not quite. As it turns out, this is the funny way in which IDL allows you to specify over which dimension to concatenate arrays. An example is worth 42 words:

```
IDL> a=make_array(4,4,VALUE=1b)
IDL> print,a
  1   1   1   1
  1   1   1   1
  1   1   1   1
  1   1   1   1
IDL> b=make_array(4,4,VALUE=2b)
IDL> print,b
  2   2   2   2
  2   2   2   2
  2   2   2   2
  2   2   2   2
IDL> print,[a,b]
  1   1   1   1   2   2   2   2
  1   1   1   1   2   2   2   2
  1   1   1   1   2   2   2   2
  1   1   1   1   2   2   2   2
IDL> print,size([a,b],/DIMENSIONS)
         8           4
IDL> print,[[a],[b]]
  1   1   1   1
  1   1   1   1
  1   1   1   1
  1   1   1   1
  2   2   2   2
  2   2   2   2
```

```
  2   2   2   2
  2   2   2   2
IDL> print,size([[a],[b]],/DIMENSIONS)
         4         8
IDL> print,[[[a]],[[b]]]
  1   1   1   1
  1   1   1   1
  1   1   1   1
  1   1   1   1

  2   2   2   2
  2   2   2   2
  2   2   2   2
  2   2   2   2
IDL> print,size([[[a]],[[b]]],/DIMENSIONS)
         4         4         2
```

In the last case, that's IDL's way of showing you two different "planes" of an image cube, as can be seen from the listed dimensions (it can't very well print outwards from your screen, can it?). You see that:

"concatenation dimension"="Number of extra [] pairs" + 1

I.e. zero extra pairs means concatenate over the first dimension, 1 over the 2nd, etc. (here we follow IDL's confusing and arbitrary notion that everything is zero-offset except dimensions, which start at 1).

There's one more wrinkle worth keeping in mind. If you recall from the disussion in the first tutorial on dimensional juggling, a "shallow" dimension is a dimension of size 1. A single element vector has one shallow dimension. A scalar has no dimensions -- creating a philosophical dilemma of its own, but that is a separate story.

What use is such a meaningless entity? Well, you might also recall that IDL happily trims away *trailing* shallow dimensions in many assignments, but keeps leading shallow dimensions. We made use of this last fact to inflate vectors to arbitrary dimension in the last tutorial. Here we will make use of the symmetric corollary: IDL will also arbitrarily *create* trailing shallow dimensions for you, as needed, during concatenation. In fact, this happened in the last case above. An example:

```
IDL> print,[ [1], [2] ]
       1
       2
IDL> print,size([ [1], [2] ],/DIMENSIONS)
         1         2
```

Aha, here we have a "column vector". We simply specified concatenating over the 2nd dimension with one surrounding pair of braces. But how can this be? The scalar "1" does not have a second dimension. Luckily, IDL created it for us, out of thin air. We can use this magic-shallow-dimension-creation to scale up even further:

```
IDL> print,[ [[1]], [[2]] ]
       1

       2
IDL> print,size([ [[1]], [[2]] ],/DIMENSIONS)
          1           1           2
```

We've made a 3-d array out of two scalars! You can also of course concatenate many things together at once:

```
IDL> a=[ [[1]], [[2]], [[3]], [[make_array(1,1,2,VALUE=5)]] ]
```

A trivial but practical use for this method is appending a "row" to an array (as opposed to appending a "column", which requires the tricks of the last tutorial to create a *leading* shallow dimension):

```
IDL> array=[ [array], [indgen(10)] ]
```

Another common use is adding a plane or planes to an image cube:

```
IDL> imcube=[ [[imcube]],[[newim]],[[newim2]] ]
```

Each of the [[]] pairs says to use the 3rd dimension for concatenation, creating shallow dimensions for you first if they don't exist (as for, e.g, the 2-d arrays "newim" and "newim2"). You must use the same number of enclosing brace pairs on each element. And, all dimensions up to the one over which you are concatenating must agree.

One caveat: a bug in IDL (as I see it) limits the practical concatenation dimension to 3, even though up to 8 dimensions are supported (i.e. only two pairs of extra brackets are allowed per entry... sorry no [[[[[[[a]]]]]]] permitted). You'll need higher magic if you use 8 dimensional datasets anyway.

JD
