Subject: Re: Efficient comparison of arrays
Posted by Marty Ryba on Mon, 11 Aug 1997 07:00:00 GMT
View Forum Message <> Reply to Message

This is a multi-part message in MIME format.

--------------66221E172010
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

David Fanning wrote:
>>> Given vectors of the type...
>>>
>>> a = [1,2,3,4,5]
>>> b = [3,4,5,6,7]
>>>
>>> What is the most efficient way to determine which values that occur in
>>> a also occur in b (i.e., the values [3,4,5] occur in both a and b).
> I don't know if this is the most efficient way (it probably isn't),
> but this is my off-the-cuff way of solving this problem.
>
> FUNCTION A_in_B, a, b
> num = N_Elements(a)
> vector = FltArr(num)
> FOR j=0,num-1 DO BEGIN
> index = Where(a(j) EQ b, count)
> IF count GT 0 THEN vector(j) = 1 ELSE vector(j)=0
> ENDFOR
> solution = a(Where(vector EQ 1))
> RETURN, solution
> END
>
> When I run the example case above, I get a vector with the values
> [3,4,5].

Here's a routine, called SYNCHRONIZE, that I whipped up thanks to some help from RSI. I had the problem of two data streams, each with a structure with a tag called "dwell" that needed matching up. This routine works great except possibly for some funny behavior if a given number is repeated in a data stream (nah, that shouldn't happen in real data). If you don't need the full-blown procedure, the guts of the algorithm should be obvious.

--
Dr. Marty Ryba | Of course nothing I say here is official
MIT Lincoln Laboratory | policy, and Laboratory affililaton is
ryba@ll.mit.edu | for identification purposes only,
| blah, blah, blah, ...

----------66221E172010
Content-Type: text/plain; charset=us-ascii; name="synchronize.pro"
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="synchronize.pro"

```
;+
; Name:
; SYNCHRONIZE
; Purpose:
; Match up two arrays of structures by a tag name
; Usage:
; synchonize,a,b[,ause=ause][,buse=buse][,keywords=values]
; Inputs:
; A & B are arrays of structures to be sychronized.  They are
; returned containing only those members that match within the
; specified tolerance for the tag fields being matched.
; Optional Keyword Inputs:
; tags - String array containing the tag names to match by.  Defaults
;  to ['dwell','dwell'].  Case insensitive.
; tolerance - Maximum difference to declare a match.  All comparisions
;  are double precision.  Defaults to 0.0 (exact match).
; Optional Outputs:
; ause - Set of array indicies used to convert input A to output A.
;  Useful if you have 2 already synchonized arrays and need to
;  synch a third.
; buse - Same for B
; Restrictions:
;     The structures should have only one member with the tag name given.
; If there are multiple structure members with the same tag, SYNCHRONIZE
; will use the first.
;     Cannot use tags from nested structures.
; Modification History:
; M.F. Ryba, MIT/LL, June 93, Created from algorithm written by David
;  Stern of RSI.
; M.F. Ryba, Jan 95, added TEMPORARY and check for whether the
;  structure is shortened.
;-
PRO Synchronize, a, b, ause=ause, buse=buse, tags=tags, tolerance=tolerance, $
                help=help

IF keyword_set(help) THEN BEGIN
   doc_library, 'synchronize'
   return
ENDIF

IF n_elements(tags) EQ 0 THEN tags = ['dwell', 'dwell']
IF n_elements(tolerance) EQ 0 THEN tolerance = 0.0d
```

```
tags = strupcase(tags)
atags = tag_names(a)
btags = tag_names(b)

ai = where(atags EQ tags(0), cnt)
IF cnt EQ 0 THEN BEGIN
   string = 'Tag '+tags(0)+' not found in structure A'
   message, string
ENDIF
IF cnt GT 1 THEN BEGIN
   string = 'Structure A has more than 1 tag named '+tags(0)+ $
     '; will use the first'
   message, string, /informational
ENDIF

bi = where(btags EQ tags(1), cnt)
IF cnt EQ 0 THEN BEGIN
   string = 'Tag '+tags(1)+' not found in structure B'
   message, string
ENDIF
IF cnt GT 1 THEN BEGIN
   string = 'Structure B has more than 1 tag named '+tags(1)+ $
     '; will use the first'
   message, string, /informational
ENDIF

ai = ai(0) & bi = bi(0)
tmp = [double(a.(ai)), double(b.(bi))]
sortab = sort(tmp)
tmp = tmp(sortab)

match = where((tmp(1:*) - tmp) LE tolerance, cnt)

IF cnt GT 0 THEN BEGIN
   aeq = sortab(match)
   beq = sortab(match+1)
   tmp = aeq < beq
   beq = (beq > aeq) - n_elements(a)
   aeq = tmp
   IF n_elements(aeq) NE n_elements(a) THEN a = (temporary(a))(aeq)
   IF n_elements(beq) NE n_elements(b) THEN b = (temporary(b))(beq)
   ause = aeq
   buse = beq
ENDIF ELSE BEGIN
   print, 'No matches found between A.'+tags(0)+' and B.'+tags(1)
   ause = -1
   buse = -1
ENDELSE
```

```
return
END

--------------66221E172010--
```