Subject: Re: The intersection of 2 arrays - where_ar.pro [1/1]
Posted by David.Steele on Tue, 04 Mar 1997 08:00:00 GMT
View Forum Message <> Reply to Message

--*-*-*- Next Section -*-*-*
Content-Type: Text/Plain; charset=US-ASCII

In article <331B37EE.41C6@irc.chmcc.org>, williams@irc.chmcc.org says...
>
> I know that I can find the union of two arrays by doing something like
>
> temp = [array1, array2]
> union = temp(uniq(temp, sort(temp)))
>
> But how would I go about finding the intersection of the two arrays?
> i.e. Is there a nice vector way of doing it rather than brute force?
> (aka: Which IDL function did I miss this time?)
>
> Thanks in advance,
> Phil
> --
> /** ********/
> Phil Williams, Ph.D.
> Research Instructor
> Children's Hospital Medical Center "One man gathers what
> Imaging Research Center another man spills..."
> 3333 Burnet Ave. -The Grateful Dead
> Cincinnati, OH 45229
> email: williams@irc.chmcc.org
> URL: http://scuttle.chmcc.org/~williams/
> /** ********/

The attached routine (WHERE_ARRAY.PRO) was posted a few years back on this newsgroup, courtesy of Stephen Strebel. It uses a vector algorithm, and does what I think Phil wants to do.

Good luck!

Dave

David P. Steele Ph: (306) 966-6447
ISAS, University of Saskatchewan Fax: (306) 966-6400
116 Science Place David.Steele@usask.ca
Saskatoon SK S7N 5E2 DANSAS::STEELE
--*-*-*- Next Section -*-*-*
Content-Type: Text/Plain; charset=ISO-8859-1

;

```
;+
; NAME:
; WHERE_ARRAY
;
; PURPOSE:
; Return the indices where vector B exists in vector A.
; Basically a WHERE(B EQ A) where B and A are 1 dimensional arrays.
;
; CATEGORY:
;       Array
;
; CALLING SEQUENCE:
; result = WHERE_ARRAY(A,B)
;
; INPUTS:
; A vector that might contains elements of vector B
; B vector that we would like to know which of its
;  elements exist in A
;
; OPTIONAL INPUTS:
;
; KEYWORD PARAMETERS:
; iA_in_B  return instead the indices of A that are in
;   (exist) in B
;
; OUTPUTS:
; Index into B of elements found in vector A.  If no
; matches are found -1 is returned.  If the function is called
; with incorrect arguments, a warning is displayed, and -2 is
; returned (see SIDE EFFECTS for more info)
;
; OPTIONAL OUTPUTS:
;
; COMMON BLOCKS:
;  None
;
; SIDE EFFECTS:
; If the function is called incorrectly, a message is displayed
; to the screen, and the !ERR_STRING is set to the warning
; message.  No error code is set, because the program returns
; -2 already
;
; RESTRICTIONS:
; This should be used with only Vectors.  Matrices other then
; vectors will result in -2 being returned.  Also, A and B must
; be defined, and must not be strings!
;
; PROCEDURE:
```

```
;
; EXAMPLE:
; IDL> A=[2,1,3,5,3,8,2,5]
;       IDL> B=[3,4,2,8,7,8]
; IDL> result = where_array(a,b)
; IDL> print,result
;           0          0          2          2          3          5
; SEE ALSO:
; where
;
; MODIFICATION HISTORY:
;  Written by: Dan Carr at RSI (command line version) 2/6/94
;   Stephen Strebel  3/6/94
;    made into a function, but really DAN did all
;    the thinking on this one!
;   Stephen Strebel  6/6/94
;    Changed method, because died with Strings (etc)
;    Used ideas from Dave Landers.  Fast TOO!
;   Strebel 30/7/94
;    fixed checking structure check
;-
FUNCTION where_array,A,B,IA_IN_B=iA_in_B

; Check for: correct number of parameters
;    that A and B have each only 1 dimension
;    that A and B are defined
if (n_params() ne 2 or (size(A))(0) ne 1 or (size(B))(0) ne 1 $
 or n_elements(A) eq 0 or n_elements(B) eq 0) then begin
 message,'Inproper parameters',/Continue
 message,'Usage: result = where_array(A,B,[IA_IN_B=ia_in_b]',/Continue
 return,-2
endif

;parameters exist, let's make sure they are not structures
if ((size(A))((size(A))(0)+1) eq 8 or $
 (size(B))((size(B))(0)+1) eq 8) then begin
 message,'Inproper parametrs',/Continue
 message,'Parameters cannot be of type Structure',/Continue
 return,-2
endif

; build two matrices to compare
Na = n_elements(a)
Nb = n_elements(b)
l = lindgen(Na,Nb)
AA = A(l mod Na)
BB = B(l / Na)
```

```
;compare the two matrices we just created
I = where(AA eq BB)
Ia = i mod Na
Ib = i / na

; normally (without keyword, return index of B that
; exist in A
if keyword_set(iA_in_B) then index = Ia $
else index = Ib

;make sure a valid value was found
if Ia(0) eq -1 or Ib(0) eq -1 then index = -1

return,index

END
```

--*-*-*- Next Section -*-*-*--
