Subject: Re: HELP: Finding Array Element Positions
Posted by stl on Thu, 30 Jun 1994 06:11:43 GMT
View Forum Message <> Reply to Message

In article <772920404snz@babbage.demon.co.uk> Richard@babbage.demon.co.uk writes:
> A PV-WAVE (version 4.20 on Solaris 2.3) Array Question:
>
> I have two string arrays, e.g.
>
> STRARR (5) = ['a','b','c','d','e']
>   STRARR (2) = ['b','e']
>
> I wish to find the elements which are common, i.e the result would be
>
> INTARR (2) = [1,4]
>
> I am currently performing this in a FOR loop (sacrilege!). With WAVE's
> array manipulation I would expect there to be a more elegant solution.

Hi,

well, about a month ago a posted this exact program to the net, and
there was lots of other discussion about this problem. Here is the best
solution (from misc net discussion that I converted into a program)

 -stephen Strebel

\\\\\\\\\\\\\\\\\ cut here \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

```
;
;+
; NAME:
; where_array.pro
;
; PURPOSE:
; return the indecies of where vector B exists in vector A.
;  Basicly 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 the might contains elements of vector B
;  B vector the we would like to know which of its
;    elements exist in A
;
; OPTIONAL INPUTS:
;
; KEYWORD PARAMETERS:
;  iA_in_B  return instead the indecies 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 diplayed
;  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.  Matricies 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
;
; 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!
;
;
;-
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,[COMMON_NUM=com],[COM_DUP=dup]',/Continue
 return,-2
endif

;parameters exist, lets make sure they are not structures
if ((size(A))(3) eq 8 or (size(B))(3) eq 8) then begin
 message,'Inproper parametrs',/Continue
 message,'Parameters cannot be of type Structure',/Continue
 return,-2
endif

; build two matricies 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

\\\\\\\\\\\\\\\\\\\\\ finish cutting here \\\\\\\\\\\\\\\\\\\\\\\\

cheers.

--
Stephen C Strebel / SKI TO DIE
strebel@sma.ch / and
Swiss Meteorological Institute, Zuerich / LIVE TO TELL ABOUT IT
01 256 93 85 / (and pray for snow)
