Subject: Re: how to get index of array B data in array A
Posted by David Foster on Wed, 17 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Gary Fu wrote:
>
> Hi,
>
> Is there a simple way to get the index of the array B data in array A ?
>
> For example : A = [10,20,30,40,50], B = [10, 20, 33, 10, 30, 40, 9]
> I want like to get the index array of B in A [0, 1, -1, 0, 2, 3, -1]
>

Gary -

I have a routine FIND_ELEMENTS.PRO that is very primitive, it just uses a basic for loop to allow searches when n_elements(B) > 1. It returns the indices into A of elements of B that were found in A. You can get it from:

ftp://bial8.ucsd.edu/pub/software/idl/share

This routine allows you to adjust the elements that you find, and so is useful when you are manipulating ROI's within images.

There IS a vectorized approach that was posted some time ago by Dan Carr of RSI, called WHERE_ARRAY(), that is faster in some circumstances, but it uses a lot of memory. In our applications, A often has some 65,000 elements and B may have 20 or so, and we find FIND_ELEMENTS to be faster (note args for WHERE_ARRAY are backwards):

```
IDL> t1=systime(1) & ind=find_elements(image1, a) & print, systime(1)-t1
      0.68172407

IDL> t1=systime(1) & ind=where_array(a, img) & print, systime(1)-t1
       5.6160550
```

{A = 256x256 integer array, a brain image; B = [23,38,28,93,24,12,87,64]
}

However, as B gets larger, WHERE_ARRAY() looks better and better!

I've included WHERE_ARRAY.PRO and another file of routines that were posted by someone at RSI not too long ago. They use histograms to compare two vectors. The intersection routine returns the actual elements, not the indices, and all ignore duplicate elements. But you may find them useful, as I have in certain applications. They

are quite fast.

One of these days when I have time I'm going to write a WHERE_V()
routine in C that can be called by a wrapper.

Dave
--

~~~~~~ ~~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~ ~~~~~~~

```
;
;+
; 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

;  ____________________________________________________________________ _____
;
; SETARRAY_UTILS.PRO  [RSI]  9-04-97
;
; Routines posted on newsgroup by RSI. SetIntersection() is much
; faster than Find_Elements(), but it returns the elements
; themselves, not the indices. Also, it ignores duplicate elements.
;
;
; Set operators.  Union, Intersection, and Difference (i.e. return
; members of A that are not in B.)
;
; These functions operate on arrays of positive integers, which need
```

```
; not be sorted.  Duplicate elements are ignored, as they have no
; effect on the result.
;
; The empty set is denoted by an array with the first element equal to -1.
;
; These functions will not be efficient on sparse sets with wide
; ranges, as they trade memory for efficiency.  The HISTOGRAM function
; is used, which creates arrays of size equal to the range of the
; resulting set.

; For example:
;   a = [2,4,6,8]
;   b = [6,1,3,2]
; SetIntersection(a,b) = [ 2, 6]       ; Common elements
; SetUnion(a,b) = [ 1, 2, 3, 4, 6, 8]  ; Elements in either set
; SetDifference(a,b) = [ 4, 8]         ; Elements in A but not in B
; SetIntersection(a,[3,5,7]) = -1 = Null Set

 ;---------------------------------------------------------- --------
FUNCTION SetUnion, a, b
if a[0] lt 0 then return, b    ;A union NULL = a
if b[0] lt 0 then return, a    ;B union NULL = b
return, where(histogram([a,b], OMIN = omin)) + omin ;Return combined set
end


 ;---------------------------------------------------------- --------
FUNCTION SetIntersection, a, b

minab = min(a, MAX=maxa) > min(b, MAX=maxb) ;Only need intersection of ranges
maxab = maxa < maxb

;If either set is empty, or their ranges don't intersect: result = NULL.
if maxab lt minab or maxab lt 0 then return, -1

r = where((histogram(a, MIN=minab, MAX=maxab) ne 0) and  $
        (histogram(b, MIN=minab, MAX=maxab) ne 0), count)
if count eq 0 then return, -1 else return, r + minab
end


 ;---------------------------------------------------------- --------
FUNCTION SetDifference, a, b  ; = a and (not b) = elements in A but not in B
mina = min(a, MAX=maxa)
minb = min(b, MAX=maxb)
if (minb gt maxa) or (maxb lt mina) then return, a ;No intersection...
r = where((histogram(a, MIN=mina, MAX=maxa) ne 0) and $
        (histogram(b, MIN=mina, MAX=maxa) eq 0), count)
```

```
if count eq 0 then return, -1 else return, r + mina
end


; -------- Message from RSI to NewsGroup ------------------------------
;
; A somewhat belated reply to the numerous postings on finding the
; common elements of vectors:

; > 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).
; >

; Below appear three IDL functions that operate on sets represented by
; arrays of positive integers.  The SetIntersection(a,b) function
; returns the common elements, SetUnion(a,b) returns all unique elements
; in both arguments, and SetDifference(a,b) returns the elements
; (members) in a but not in b.

; It is faster than previously published functions, e.g. contain() and
; find_elements().

; Hope this helps,

; Research Systems, Inc.
```

File Attachments

1) where_array.pro, downloaded 73 times
2) setarray_utils.pro, downloaded 85 times