Subject: matching lists
Posted by Mark Fardal on Fri, 10 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Hi,

I have been looking at properties of particles in a simulation, and sometimes need to match up the particles in two different subsets. I typically have (quantity A, index #) for one set of particles, and (quantity B, index #) for another set, and want to compare quantities A and B for the particles that are in both sets.

As of late last night I could not think of a good way to do this; WHERE inside a for-loop would be very slow. Maybe I'm missing something easy, but in any case here's a solution inspired by the recently submitted SETINTERSECTION function. Hope somebody finds it useful.

Mark Fardal
UMass

```
;+
; NAME:
;       LISTMATCH
;
; PURPOSE:
;   find the indices where a matches b
;   works only for SETS OF UNIQUE INTEGERS, e.g., array indices
;
;   for example: suppose you have a list of people with their ages and
;   social security numbers (AGE, AGE_SS), and a partially overlapping
;   list of people with their incomes and s.s. numbers (INCOME,
;   INCOME_SS).  And you want to correlate ages with incomes in the
;   overlapping subset.  Call
;      LISTMATCH, AGE_SS, INCOME_SS, AGE_IND, INCOME_IND
;   then AGE[AGE_IND] and INCOME[INCOME_IND] will be the desired
;   pair of variables.
;
; AUTHOR:
;   Mark Fardal
;   UMass (fardal@weka.astro.umass.edu)
;
; CALLING SEQUENCE:
;   LISTMATCH, a, b, a_ind, b_ind
;
; INPUTS:
;   a and b are sets of unique integers (no duplicate elements)
```

```
;
; OUTPUTS:
;   a_ind, b_ind are the indices indicating which elements of a and b
;   are in common
;
; RESTRICTIONS:
;   if the indices are not unique some matching elements may be skipped...
;   or is it worse than that?
; EXAMPLE:
;   a = [2,4,6,8]
;   b = [6,1,3,2]
;   listmatch, a, b, a_ind, b_ind
;    print, a[a_ind]
;       2       6
;    print, b[b_ind]
;       2       6
;
;
; MODIFICATION HISTORY:
;   none
; BUGS:
;   tell me about them
; ACKNOWLEDGEMENTS:
;   trivial modification of SETINTERSECTION from RSI
;
pro listmatch, a, b, a_ind, b_ind
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 (which is denoted by integer = -1)
if maxab lt minab or maxab lt 0 then begin
  a_ind = -1
  b_ind = -1
  return
endif

ha = histogram(a, MIN=minab, MAX=maxab, reverse_indices=reva)
hb = histogram(b, MIN=minab, MAX=maxab, reverse_indices=revb)

r = where((ha ne 0) and (hb ne 0), count)
if count gt 0 then begin
  a_ind = reva[reva[r]]
  b_ind = revb[revb[r]]
  return
endif else begin
  a_ind = -1
  b_ind = -1
  return
```

```
endelse

end
```

Subject: Re: matching lists
Posted by Dick Jackson on Wed, 06 Apr 2005 20:58:08 GMT
View Forum Message <> Reply to Message

Hi all,

This is a follow-up to a thread from March, 2000, where JD Smith had offered
functions for returning indices of the values in one array that are found in
a second array. Perhaps due to a change in what Sort() does with identical
entries, the "ind_int_SORT" function gave erroneous results.

(I'm on IDL 6.1.1, Windows XP Pro, and I get this:

```
 IDL> print,sort([1,1])
           1           0
)
```

Here's a simple fix that lets it work no matter *what* order Sort() puts the
indices of identical entries in. (even if it is inconsistent within a single
call to Sort()!)

```
=====
;; Return the indices of values in a which exists anywhere in b
;; (one only for repeated values)
function ind_int_SORT, a, b
    flag=[replicate(0b,n_elements(a)),replicate(1b,n_elements(b) )]
   s=[a,b]
   srt=sort(s)
   s=s[srt] & flag=flag[srt]
   wh=where(s eq shift(s,-1) and flag ne shift(flag, -1),cnt)
   if cnt eq 0 then return, -1
   return,srt[wh+flag[wh]]
end
=====
```

The fix is just in the "return..." line, which had read:

```
   return,srt[wh]
```

... and which for me returned:

```
IDL> print,ind_int_SORT(['1','2'],['1','2'])
          2           3
```

Here we need to know whether an item in 'srt' came from 'a' or 'b', and we
know that from 'flag'. So now, we get:

```
IDL> print,ind_int_SORT(['1','2'],['1','2'])
           0           1
```

Note that this doesn't guarantee that the result indices are sorted:

```
IDL> print,ind_int_SORT(['1','1','2','3'],['1','2','4'])
           1           0           2
```

Hmm, since JD's always did guarantee this (like Where() does it), here's one more amendment:

```
=====
;; Return the indices of values in a which exists anywhere in b
;; (one only for repeated values)
function ind_int_SORT, a, b
    flag=[replicate(0b,n_elements(a)),replicate(1b,n_elements(b) )]
   s=[a,b]
   srt=sort(s)
   s=s[srt] & flag=flag[srt]
   wh=where(s eq shift(s,-1) and flag ne shift(flag, -1),cnt)
   if cnt eq 0 then return, -1
   result=srt[wh+flag[wh]]
   return,result[sort(result)]
end
=====
```

Mark Fardal posted a version of this with a different calling sequence, here's the amended version of that (two changes, for a_ind and b_ind):

```
=====
pro listmatch, a, b, a_ind, b_ind
   flag=[replicate(0b,n_elements(a)),replicate(1b,n_elements(b) )]
  s=[a,b]
  srt=sort(s)
  s=s[srt] & flag=flag[srt]
  wh=where(s eq shift(s,-1) and flag ne shift(flag, -1),cnt)
  if cnt ne 0 then begin
    a_ind = srt[wh+flag[wh]]
    a_ind = a_ind[sort(a_ind)]
    b_ind = srt[wh+1-flag[wh]]-N_Elements(a)
    b_ind = b_ind[sort(b_ind)]
  endif else begin
    a_ind = -1
    b_ind = -1
    return
  endelse
end
=====
```

Thanks to JD and Mark for their original work!

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

Subject: Re: matching lists
Posted by JD Smith on Wed, 06 Apr 2005 23:44:05 GMT
View Forum Message <> Reply to Message

On Wed, 06 Apr 2005 20:58:08 +0000, Dick Jackson wrote:

```
> Hi all,
>
> This is a follow-up to a thread from March, 2000, where JD Smith had offered
> functions for returning indices of the values in one array that are found in
> a second array. Perhaps due to a change in what Sort() does with identical
> entries, the "ind_int_SORT" function gave erroneous results.
>
> (I'm on IDL 6.1.1, Windows XP Pro, and I get this:
>  IDL> print,sort([1,1])
>           1           0
> )
>
```

I'm on 6.1.1 Linux, and I get:

```
IDL> print,sort([1,1])
           0           1
```

Anybody else get Dick's behavior?

```
> Here's a simple fix that lets it work no matter *what* order Sort() puts the
> indices of identical entries in. (even if it is inconsistent within a single
> call to Sort()!)
>
> =====
> ;; Return the indices of values in a which exists anywhere in b
> ;; (one only for repeated values)
> function ind_int_SORT, a, b
>     flag=[replicate(0b,n_elements(a)),replicate(1b,n_elements(b) )]
>    s=[a,b]
>    srt=sort(s)
>     s=s[srt] & flag=flag[srt]
>     wh=where(s eq shift(s,-1) and flag ne shift(flag, -1),cnt)
>     if cnt eq 0 then return, -1
>     result=srt[wh+flag[wh]]
>     return,result[sort(result)]
>  end
>  =====
```

Interesting. Although I knew depending on the SORT order was an issue, I didn't think it would actually show up. Your changes make my SORT based matcher somewhat closer to the MATCH routine written by Don Lindler way back in the 1986, as part of the AstroLib library. In that same thread you mentioned (this thread, I guess!), I had commented:

By the way, I found an implementation I had mentioned a while back on the news group but had forgotten about from the nasa lib called "match" which does pretty much the same thing. It's probably less efficient, since it uses an auxiliary list of indices in addition to the flag vector, instead of just using the sort() results directly, and performs a few more "where" tests as a result. But a similar idea, written first in 1986! Match() is also more immune to changes in sort() than my routine, as a result of carrying around these additional index arrays.

Thanks,

JD

Subject: Re: matching lists
Posted by David Fanning on Thu, 07 Apr 2005 01:07:40 GMT

View Forum Message <> Reply to Message

JD Smith writes:

```
>  I'm on 6.1.1 Linux, and I get:
>
>  IDL> print,sort([1,1])
>            0          1
>
>  Anybody else get Dick's behavior?
```

I'm guessing the Astronomy Library routine uses BSORT, avoiding the SORT problem altogether.

Cheers,

David

P.S. Microsoft is notorious for this SORT problem, is
what I hear.
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: matching lists
Posted by JD Smith on Thu, 07 Apr 2005 01:42:27 GMT
View Forum Message <> Reply to Message

On Wed, 06 Apr 2005 19:07:40 -0600, David Fanning wrote:

```
> JD Smith writes:
>
>> I'm on 6.1.1 Linux, and I get:
>>
>> IDL> print,sort([1,1])
>>            0           1
>>
>> Anybody else get Dick's behavior?
>
> I'm guessing the Astronomy Library routine uses BSORT,
> avoiding the SORT problem altogether.
>
> Cheers,
>
> David
>
> P.S. Microsoft is notorious for this SORT problem, is
> what I hear.
```

Well, technically either answer is correct, since sorting two equal
elements can be done in either order. MATCH uses regular SORT, but just
keeps it's own index lists to avoid this problem.

JD