Subject: Re: matching lists
Posted by Mark Fardal on Mon, 13 Mar 2000 08:00:00 GMT
View Forum Message <> Reply to Message

Hi,

Somehow I knew that if I mentioned HISTOGRAM, that would stir up some
real IDL programmers. :->

J.D.'s sort method seems like the winner. The modest speed advantage
of the histogram method in certain cases is not important. If you are
in a situation where just matching up the elements is the limitation,
you are probably going to be in trouble doing any analysis with them
(let alone reading them in).

The problem of repeated elements, which is the only advantage of
WHERE_ARRAY, is not of any concern, at least to me. The point of
the key variables a and b is that they are supposed to be unique
identifiers. I would just like the routine not to break completely
in case the same element was copied into the arrays twice. The
sort method does fine in that respect (finds the last of the
duplicate elements in a and the first in b).

The only flaw with the sort method is that sooner or later RSI is
going to break its own SORT function, just like it does with all of
its other code...

> The standard where_array, as posted a few years back, and modified slightly for
> the case of the null intersection, is attached. It will work with floating
> point and other data types also. It works by inflating the vectors input to 2-d
> and testing for equality in one go. It will also handle the case of repeated
entries.

Hope WHERE_ARRAY does not become "standard", since it's clearly inferior
to the sort method.

For completeness, using the sort method inside the calling sequence
I originally posted would look like:

```
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]
   b_ind = srt[wh+1] - n_elements(a)
```

```
  endif else begin
    a_ind = -1
    b_ind = -1
    return
  endelse
end
```

Mark Fardal
UMass
