Subject: Re: Efficient comparison of array location in two lists
Posted by wlandsman on Tue, 22 Feb 2011 00:43:12 GMT
View Forum Message <> Reply to Message

On Monday, February 21, 2011 6:59:57 PM UTC-5, Matt Francis wrote:

>
> IDL_MAGIC(foo1,foo2,indx1,indx2)
> print,indx1,indx2
> -> [0,1,2] [0,2,1]

One procedure to do this is match.pro
(http://idlastro.gsfc.nasa.gov/ftp/pro/misc/match.pro).

the calling sequence would be
IDL> match,foo1,foo2,indx1,indx2

--Wayne

Subject: Re: Efficient comparison of array location in two lists
Posted by Heinz Stege on Tue, 22 Feb 2011 02:45:50 GMT
View Forum Message <> Reply to Message

On Mon, 21 Feb 2011 15:59:57 -0800 (PST), Matt Francis wrote:

> I have two sets of data, both comprised of a STRARR labelling the data
> and a FLTARR of the data itself.
>
> I want to compare the data in the two sets, but the order and number
> of labels are different, so I'd first need to to find which array
> elements in each FLTARR correspond to the same label.
>
> It's trivial to see how this can be accomplished by simply looping
> over either data set and using WHERE(STRCMP()), but is there a way to
> do this without the loop?
>
> To make it clear, say I had:
>
> foo1 = ['a','b','c']
> foo2 = ['a','c','b']
>
> I want a single line that produces:
>
> IDL_MAGIC(foo1,foo2,indx1,indx2)
> print,indx1,indx2
> -> [0,1,2] [0,2,1]
>

> So that could compare
>
> foo1_data[indx1] and foo2_data[indx2] in whatever way I wanted.
>
> Any thoughts?

You can manage it in a short and very fast way by use of the built-in
function value_locate:

```
indx1=sort(foo1)
if n_elements(foo1) ge 2 then $
  indx2=value_locate(foo1[indx1],foo2) $
else $
  ; value locate does not work for vectors with 1 element.
  ; It would be nice, if ITT VIS could make it working in IDL 8.1.
  indx2=lonarr(n_elements(foo2))
ii=where(foo1[indx1] eq foo2[indx2],count)
if count le 0 then message,'No matching labels found.'
indx1=indx1[ii]
indx2=indx2[ii]
```

The labels foo1 as well as foo2 are assumed to be unique.

I believe, value_locate is an often underestimated function.

Heinz

Subject: Re: Efficient comparison of array location in two lists
Posted by David Fanning on Tue, 22 Feb 2011 03:17:14 GMT
View Forum Message <> Reply to Message

Heinz Stege writes:

> I believe, value_locate is an often underestimated function.

And there is a understated statement! :-)

Let's just say a lot of neat functionality in the new
book wouldn't be possible without the power
of Value_Locate!

Cheers,

David

--

David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Efficient comparison of array location in two lists
Posted by wlandsman on Tue, 22 Feb 2011 04:02:30 GMT
View Forum Message <> Reply to Message

On Monday, February 21, 2011 9:45:50 PM UTC-5, Heinz Stege wrote:

> The labels foo1 as well as foo2 are assumed to be unique.

I believe that this method also requires that foo2 be a one-to-one mapping of the elements in foo1, and fails for example, to find the matching 'e' values in the two vectors:

foo1 = ['b','c','d','e']
foo2 = ['a','e','f','g']

> I believe, value_locate is an often underestimated function.

And even some of us who use it often, forget that it can be used with strings. --Wayne

---

Subject: Re: Efficient comparison of array location in two lists
Posted by Jeremy Bailin on Tue, 22 Feb 2011 05:21:06 GMT
View Forum Message <> Reply to Message

> I believe, value_locate is an often underestimated function.

You won't hear any arguments from me on that. :-)=

-Jeremy.

Subject: Re: Efficient comparison of array location in two lists
Posted by Heinz Stege on Tue, 22 Feb 2011 10:16:22 GMT
View Forum Message <> Reply to Message

On Mon, 21 Feb 2011 20:02:30 -0800 (PST), wlandsman wrote:

> On Monday, February 21, 2011 9:45:50 PM UTC-5, Heinz Stege wrote:
>
>> The labels foo1 as well as foo2 are assumed to be unique.
>
> I believe that this method also requires that foo2 be a one-to-one mapping of the elements in

foo1, and fails for example, to find the matching 'e' values in the two vectors:
>
> foo1 = ['b','c','d','e']
> foo2 = ['a','e','f','g']
>
>> I believe, value_locate is an often underestimated function.
>
> And even some of us who use it often, forget that it can be used with strings. --Wayne

Good morning!

I should not answer to postings in the middle of the night. I copied
the commands from an existing program in my library and made mistakes
addopting them to the OPs question. Thank you Wayne, for bringing it
out.

And here is the correction:

```
indx1=sort(foo1)
if n_elements(foo1) ge 2 then $
   indx1=indx1[value_locate(foo1[indx1],foo2)] $
else $
   indx1=0
indx2=where(foo1[indx1] eq foo2,count)
if count le 0 then message,'No matching labels found.'
indx1=indx1[indx2]
```

Note, that you have to make changes, if you set the strictarrsubs or
idl2 compiler option. For this case:

```
indx1=sort(foo1)
if n_elements(foo1) ge 2 then $
   indx1=indx1[value_locate(foo1[indx1],foo2)>0] $
else $
   indx1=lonarr(n_elements(foo2))
indx2=where(foo1[indx1] eq foo2,count)
if count le 0 then message,'No matching labels found.'
indx1=indx1[indx2]
```

I hope, that it is working now. Sorry to everybody.

Heinz

Subject: Re: Efficient comparison of array location in two lists
Posted by Heinz Stege on Tue, 22 Feb 2011 10:18:48 GMT
View Forum Message <> Reply to Message

The code above contains errors. Wayne Landsman shows an example in another thread. Here is a copy from my correction:

Good morning!

I should not answer to postings in the middle of the night. I copied the commands from an existing program in my library and made mistakes addopting them to the OPs question. Thank you Wayne, for bringing it out.

And here is the correction:

```
indx1=sort(foo1)
if n_elements(foo1) ge 2 then $
   indx1=indx1[value_locate(foo1[indx1],foo2)] $
else $
   indx1=0
indx2=where(foo1[indx1] eq foo2,count)
if count le 0 then message,'No matching labels found.'
indx1=indx1[indx2]
```

Note, that you have to make changes, if you set the strictarrsubs or idl2 compiler option. For this case:

```
indx1=sort(foo1)
if n_elements(foo1) ge 2 then $
   indx1=indx1[value_locate(foo1[indx1],foo2)>0] $
else $
   indx1=lonarr(n_elements(foo2))
indx2=where(foo1[indx1] eq foo2,count)
if count le 0 then message,'No matching labels found.'
indx1=indx1[indx2]
```

I hope, that it is working now. Sorry to everybody.

Heinz
