
Subject: Re: Search single column of array - removing nasty loop
Posted by [Yngvar Larsen](#) on Thu, 01 Dec 2011 12:46:38 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Dec 1, 1:10 pm, Rob <rl...@le.ac.uk> wrote:

> On Dec 1, 12:00 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:

>

>

>

>

>

>

>

>

>

>> On Dec 1, 11:37 am, Rob <rl...@le.ac.uk> wrote:

>

>>> On Nov 30, 8:15 pm, Yngvar Larsen <larsen.yng...@gmail.com> wrote:

>

>>>> On Nov 29, 6:53 pm, Heinz Stege <public.215....@arcor.de> wrote:

>

>>>> > Hi Rob,

>

>>>> > no loop necessary:

>

>>>> > array=(randomu(seed,2,6,360,42)-.1)>0. ; sample array

>>>> > array=reform(array,n_elements(array)/42,42,/overwrite)

>>>> > ii=where(min(array,dim=2) eq 0.,count)

>>>> > if count ge 1 then array[ii,*]=0.

>>>> > array=reform(array,2,6,360,42,/overwrite)

>

>>>> Hm. The /OVERWRITE keyword to REFORM was new to me. Thanks!

>

>>>> Silly me. I have somehow always imagined that the compiler was smart

>>>> enough to do this (i.e. not copy any data, only alter the internal IDL

>>>> descriptor of the ARRAY variable) automatically when input and output

>>>> to REFORM is the same variable. But a bit of profiling shows this is

>>>> not at all the case. This will be _very_ useful many places in my

>>>> operational code...

>

>>>> A small comment to the code above: "where(min(array,dim=2) eq 0.)"

>>>> obviously only works if array contains only non-negative data. If not,

>>>> "where(total(array eq 0, 2) gt 0)" will do the trick also for floating

>>>> point data containing negative numbers, with more or less the same

>>>> performance.

>

>>>> --

>>>> Yngvar

```

>
>>> Thanks, that explains why a few results were coming out slightly
>>> differently as there are a few negative values.
>
>>> Also, the code fails when the final column only has 1 element in it.
>
>>> IDL> help, array
>>> ARRAY      DOUBLE   = Array[4320, 1]
>>> IDL> help, total(array eq 0, 2)
>>> % TOTAL: For input argument <BYTE   Array[4320]>, Dimension must be
>>> 1.
>
>> If the final column has only 1 element, the operation is not necessary
>> at all since all elements are already 0 :)
>
>> IDL sometimes behaves rather idiotic with singleton dimensions:
>
>> IDL> help, fltarr(4320, 1)
>> <Expression>  FLOAT   = Array[4320]
>
>> This is a problem when arrays are expected to be 2D, and suddenly are
>> automatically 1D. You can avoid it by adding an explicit REFORM
>> statement at the appropriate place in the code:
>
>> ;; Force ARRAY to be 2D always
>> if (size(array, /n_dimensions) eq 1) then $
>>   array = reform(array, n_elements(array), 1, /overwrite)
>
>> --
>> Yngvar
>
> I'm not sure if that's the solution as the array was already 2D:
>
>
>
>
>
>
>
>>> IDL> help, array
>>> ARRAY      DOUBLE   = Array[4320, 1]

```

Right. I suspected something like that. That's why I qualified it with
"...at the appropriate place in the code" :)

Your problem is this rather strange behavior:

```

-----
IDL> help, array

```

```
ARRAY      FLOAT   = Array[4320, 1]
IDL> help, array eq 0
<Expression>  BYTE   = Array[4320]
-----
```

So the solution is:

```
;;...
tmp = array eq 0
;; Force TMP to be 2D always
if (size(tmp, /n_dimensions) eq 1) then $
    tmp = reform(tmp, n_elements(tmp), 1, /overwrite)
ii = where(total(tmp, 2) gt 0, count)
;;...
```

--

Yngvar
