

---

Subject: Re: Not where

Posted by [Martin Schultz](#) on Mon, 07 Feb 2000 08:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Alex Schuster wrote:

>  
> David Fanning wrote:  
>  
>> Kenneth P. Bowman (bowman@null.edu) writes:  
>>  
>>> I often find myself using WHERE to divide an array into two parts. I do  
>>> one operation on the first part and a different operation on the second  
>>> part.  
>>>  
>>> It would be nice to have an auxiliary array containing all the indices  
>>> that are \*not\* returned by WHERE in i. [...]  
>  
> Easy, yes, maybe not too fast, and not very elegant. Martin Schulz wrote  
> (and probably posted) the routine INV\_INDEX, which I attached.

I'm listening ;-)

> [...]  
> This would add 1 to a[i] and subtract 5 from a[NOT(i)], for example. But  
> I admit that  
>     j = inv\_index( i )  
>     a[i] = a[i] + 1  
>     a[j] = a[j] - 5  
> looks better. (Not cooler, but better.)

It might not work, though ;-(

1. You \*must\* supply the second parameter (totaln), e.g. as in  
   j = inv\_index(i,n\_elements(data))
2. I had to fix a bug which was that I used short ints instead of long.

So, please find the new version of this routine attached ...

Cheers,  
Martin

--

```
[[ Dr. Martin Schultz  Max-Planck-Institut fuer Meteorologie  [[  
[[           Bundesstr. 55, 20146 Hamburg           [[  
[[           phone: +49 40 41173-308           [[  
[[           fax: +49 40 41173-298           [[
```

```

[[ martin.schultz@dkrz.de                                     ]]
[[
$Id: inv_index.pro,v 1.11 1999/05/20 16:15:49 mgs Exp $
;-----
;+
; NAME:
;   INV_INDEX
;
; PURPOSE:
;   find the indices that do NOT match a WHERE condition
;
; CATEGORY:
;   array index handling
;
; CALLING SEQUENCE:
;   RESULT = INV_INDEX(INDEX,TOTALN)
;
; INPUTS:
;   INDEX : an index array, e.g. previously generated by a
;           WHERE command (may be -1)
;   TOTALN : the number of elements in the reference data
;            set, i.e. totaln = n_elements(index)+n_elements(result)
;
; KEYWORD PARAMETERS:
;
; OUTPUTS:
;   an integer array with all indices that were NOT in index
;   or -1 if index was complete
;
; SUBROUTINES:
;
; REQUIREMENTS:
;
; NOTES:
;   The function returns -1 if one of the following errors occurs:
;   - invalid number of arguments
;   - index variable is undefined
;   - totaln is less than n_elements(index)
;   - totaln less or equal 1, i.e. no associated data
;   The last error does not produce an error message, since this
;   feature was found to be very useful (in EXPLORE, the widget based
;   interactive data explorer)
;
; EXAMPLE:
;   data = findgen(50)
;   index = where(data ge 25)
;   invers = inv_index(index,n_elements(data))
;   print,invers

```

```

;
; IDL prints numbers 0 through 24
;
;
; MODIFICATION HISTORY:
; mgs, 10 May 1997: VERSION 1.00
; mgs, 18 Aug 1997: - added template and check if n_elements(index) eq 0
; mgs, 05 Apr 1999: - bug fix: needed to make sure result is type long
;
;
;-
; Copyright (C) 1997, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine inv_index"
;-----

```

```
function inv_index,index,totaln
```

```
newindex = -1L ; default: nothing left
```

```
; check for errors:
```

```
if (N_Params() ne 2) then begin
  print,'INV_INDEX: wrong number of arguments'
  return,newindex
endif
```

```
if (n_elements(index) eq 0) then begin
  print,'INV_INDEX: no valid index passed'
  return,newindex
endif
```

```
if (totaln lt n_elements(index)) then begin
  print,'INV_INDEX: totaln lt n_elements(index)'
  return,newindex
endif
```

```
if (totaln le 1) then return,newindex ; no data there
```

```
; and handle the two situations:
```

```
if (max(index) lt 0) then begin ; no valid index passed
  newindex = lindgen(totaln) ; create an integer array
  return,newindex ; with totaln elements
endif
```

```
; else a valid indexarray was passed and we can construct the inverse
newindex = lindgen(totaln)
```

```
newindex(index) = -1
i = where(newindex ge 0,count)
if (count gt 0) then newindex = newindex(i) $
else newindex = -1L
```

```
return, newindex
end
```

## File Attachments

---

1) [inv\\_index.pro](#), downloaded 148 times

---