Subject: Re: remove duplicates WITHOUT sorting
Posted by btt on Mon, 12 Feb 2007 19:23:51 GMT
View Forum Message <> Reply to Message

Brian Larsen wrote:
> In the spirit of histogram has anyone written code to turn the
> REVERSE_INDICES stuff (which I can never seem to really get my head
> around) into something less confusing like a really helpful structure
> with tags i_vec and o_vec?
>
> Just asking before I reinvent the wheel next time I need to use the
> REVERSE_INDICES.
>

Hi,

Geez, I know just what you mean. I put the whole mess into one function
and let it do the thinking. You might try the following which is just a
cooked down version of what is shown in the online help...

```
***BEGIN
;+
; NAME:
;   hist_index
;
; PURPOSE:
;   Use this routine to extract the ith bin elements in
;   REVERSE_INDICES as returned by the HISTOGRAM function.
;
; CALLING SEQUENCE:
;   result = hist_index(reverse_indices, i, [chunksize = value],[count =
variable])
;
; RETURNED VALUE: (lifted from IDL online manual)
;   Returns subscripts of the original array elements
;   falling in the ith bin.
;
;  If none are in that bin then -1 is returned.  Use COUNT to check.
;
; ARGUMENTS:
;  REVERSE_INDICES  See HISTOGRAM function description.
;      You may supply this as a vector or a pointer to a vector.
;   I set this equal to the ith bin from which to return the indices.
This maybe
;      a vector of bins (contiguous or otherwise).
;
; KEYWORDS:
;   CHUNKSIZE  Set equal to the number of elements used in each
;       chunking operation (default = 10000)  This is useful
;       for very large arrays.  Memory is allocated in chunks.
;       This maybe temporarily increased to more than chuncksize if
required.
;   COUNT  Set equal to a named variable to return the
;       number of elements in the returned value.
;       Set to 0 if returned value is -1.
;
; EXAMPLE:
;   get all of the pixels in the 10th bin
;IDL> img=read_png( filepath('mineral.png', SUBDIR=['examples','data']))
;IDL>  h = histogram(img, reverse_indices = r)
;IDL> index = hist_index(r,10, count= cnt)
;IDL> help, cnt, index
;CNT             LONG      =           25
;INDEX           LONG      = Array[25]
;
; COMMENT:
;  Take care to consider the number of bins available in histogram.
;  For example, in the example above there are 255 bins (it's a byte image)
;   but it is possible to get a real (but meaningless) result.
;   In the example below, fictious bin 300 is requested.
;IDL> index = hist_index(r,300, count= cnt)
;IDL> help, cnt, index
;CNT             LONG      =         2800
;INDEX           LONG      = Array[2800]
;
; MODIFICATION HISTORY;
; 25 MAY 2004, BT written
; 22 OCT 2004 BT added multiple bin requests with chunking
;       added error handling
;-

FUNCTION hist_index, r, i, $
    CHUNKSIZE = chunksize, $
    COUNT = count

COMPILE_OPT IDL2
ON_ERROR, 2


    ni = n_elements(i)
    ctr = 0
    If n_elements(chunkSize) EQ 0 then CS = 10000 else CS = chunksize[0]
    currentSize = CS
    index = lonarr(CS)
```

```
   If SIZE(r,/TYPE) EQ 10 then Begin
      ; a pointer to R
   For j = 0, ni-1 Do begin
      ;data supplied as a pointer to vector
   if (*r)[i[j]] NE (*r)[i[j]+1] Then begin
      idx = (*r)[(*r)[i[j]]:(*r)[i[j]+1]-1]
      count = SIZE(idx,/N_ELEMENTS)
         ;increase the size of the indexkeeper if needed
      If (ctr + count) GT currentSize Then Begin
         index = [index, lonarr(CS > count)]
         currentSize += (CS > count)
      EndIf
      index[ctr] = idx
      ctr += count
   EndIf

   EndFor

   Endif Else Begin

   For j = 0, ni-1 Do begin
      ; data supplied as a vector
   if r[i[j]] NE r[i[j]+1] Then begin
      idx = r[r[i[j]]:r[i[j]+1]-1]
      count = SIZE(idx,/N_ELEMENTS)
         ;increase the size of the indexkeeper if needed
      If (ctr + count) GT currentSize Then Begin
         index = [index, lonarr(CS > count )]
         currentSize += (CS > count)
      EndIf
      index[ctr] = idx
      ctr += count
   EndIf

   EndFor

   EndElse

   Count = ctr
If Count GT 0 then $
   Return, index[0:ctr-1] Else $
   Return, -1
END
***END
```