Subject: Bug: HISTOGRAM with reverse indices AND NaN
Posted by Paul Krummel on Thu, 23 Dec 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Hi All,
I just submitted this as a bug report to RSI.
I use IDL 5.3 (and 5.2) on a windows NT 4 SP5 platform.
I have been using the histogram procedure with reverse_indices to perform bin averaging for quite a few years now.
Recently I had some data with NaN's in it so I implemented the NaN keyword.
I started getting screwy results. If there were a large number of NaN's my averaging routine would fall over due to an incorrect indice in the reverse_indice itself (see below).
Anyway thought you might be interested in this!
Cheers Paul

I am running IDL 5.3 on the platform mentioned above.
I have discovered what I think is a bug in the histogram function.
It occurs when using the reverse_indices keyword AND the NaN keyword.
The reverse indices that are returned are incorrect if there is missing data (NaN).
The procedure below should demonstrate this. I also tested this on an SGI running IRIX 6.5 and IDL 5.2.

```
; ++
pro hist_ri_fail
;
; ++++
; quick procedure to demonstrate where the
; histogram reverse indices fail when data
; contains NaNs. Counter not incremented
; correctly?.
;
;  PBK 23 Dec 1999.
;
; ++++
; make an array
a=findgen(100)
;
; Set every 3rd point to NaN
a[where(a mod 3 eq 0.)]=!values.f_nan
;
print,'a:',a
;
; do the histogram and return reverse indices.
count_mid=histogram(a, binsize=10, reverse_indices=r, $
```

```
                min=0., max=99., /NaN)
;
; ++++
; find number of Nan's and print some values
zz=where(finite(a,/nan), cnt_nan)
print,'cnt nan:',cnt_nan
print,'cnt mid:',count_mid
print,'n rev ind:',n_elements(r)
print,'rev ind:',r
;
; ++++
end
; ++

a:       NaN     1.00000    2.00000      NaN    4.00000
   5.00000       NaN    7.00000    8.00000       NaN    10.0000
   11.0000       NaN    13.0000    14.0000       NaN    16.0000
   17.0000       NaN    19.0000    20.0000       NaN    22.0000
   23.0000       NaN    25.0000    26.0000       NaN    28.0000
   29.0000       NaN    31.0000    32.0000       NaN    34.0000
   35.0000       NaN    37.0000    38.0000       NaN    40.0000
   41.0000       NaN    43.0000    44.0000       NaN    46.0000
   47.0000       NaN    49.0000    50.0000       NaN    52.0000
   53.0000       NaN    55.0000    56.0000       NaN    58.0000
   59.0000       NaN    61.0000    62.0000       NaN    64.0000
   65.0000       NaN    67.0000    68.0000       NaN    70.0000
   71.0000       NaN    73.0000    74.0000       NaN    76.0000
   77.0000       NaN    79.0000    80.0000       NaN    82.0000
   83.0000       NaN    85.0000    86.0000       NaN    88.0000
   89.0000       NaN    91.0000    92.0000       NaN    94.0000
   95.0000       NaN    97.0000    98.0000       NaN
cnt nan:          34
cnt mid:           6         7         7         6         7
           7         6         7         7         6
n rev ind:          77
rev ind:          11        51        24        31        37
           44        51        57        64        71        77
            0         1         2         3         4         5
           10        11        13        14        16        17
           19        21        24        27        30        33
           36        39        42        45        48        51
           54        57        60        63        66        69
           72        75        78        81        84        87
           90        93        96        99        61        62
           64        65        67        68        70        71
           73        74        76        77        79        80
           82        83        85        86        88        89
           91        92        94        95        97        98
```

From the output you will see that the reverse indices are not correct and quite screwy!
The second number of the reverse indices should be 17 not 51 (17+34), so the count of the number of NaN's has been added to this second indice. The rest of the pointer numbers (first 11 elements of r for this case) look fine.
The first 6 actual indices (r[11:16]) are wrong, it appears to be just 0 to 5!
The next 7 indices (r[17:23]) are correct!
Then, most of the NaN indices are listed (r[24:50], 50=24+34-7-1).
The rest of the indices are correct.
There is no way to recover all the correct indices from this. The output from histogram itself (count_mid in the example) appears to be fine. The toal number of reverse indices (77) is also correct, but as shown above the indices themselves are incorrect.

Cheers Paul

~- ~-~-~-~
Paul Krummel
CSIRO Atmospheric Research - GASLAB
Private Bag #1 Aspendale Victoria 3195 Australia
e-mail: paul.krummel@dar.csiro.au   www: http://www.dar.csiro.au/
tel: +61 3 9239 4568   fax: +61 3 9239 4444
~- ~-~-~-~