Subject: Re: histogram question
Posted by John-David T. Smith on Wed, 08 Aug 2001 22:52:26 GMT
View Forum Message <> Reply to Message

"Gregory Y. Prigozhin" wrote:
>
> Folks,
> I am sure this problem must have an elegant solution
> that is not obvious to me:
>
> I have an array X. I need to make a histogram and throw away elements
> of the array with a high count rate, say with count rate above 5 times
> median count rate.
> Brute force way is ugly and inefficient when array is not small:
>
> plothist,X,xhist,yhist
> bad=xhist[where(yhist gt 5*median(yhist),count)]
> if count ne 0 then begin
> for ind=0,count-1 do begin
> X=X[where(X ne bad[ind])]
> endfor
> endif
>
> Any suggestions?

If I understand you correctly, then REVERSE_INDICES is your friend.

Try something like:

```
h=histogram(x,binsize=1,reverse_indices=r)
bad=where(h gt 5*median(h),cnt)
if cnt eq 0 then return

;; straightforward approach
for i=0,cnt-1 do begin
  low=r[r[bad[i]] & n=r[r[bad[i]+1]]-low
  inds=indgen(n)+low
  if n_elements(list) eq 0 then list=[inds] else list=[list,inds]
endfor
```

now you have the list of bad indices into X in hand, to perform whatever
punishment is necessary.

This brings up an interesting sub-problem though. If you have a list
which consists of a series of pairs of indices, e.g.:

[1,5,7,12,15,18]

where each pair is intended to expand to the range within that pair:

[1,2,3,4,5,7,8,9,10,11,12,15,16,17,18]

how can you turn the former into the latter without a loop? This is somewhat similar to Pavel's running chunk index problem earlier in the year. Finding an answer is not trivial. It would apply directly to this problem, where the pairs are adjacent elements in the reverse indices vector. Any takers?

JD

Subject: Re: histogram question
Posted by billb on Thu, 09 Aug 2001 15:26:53 GMT
View Forum Message <> Reply to Message

> how can you turn the former into the latter without a loop? This is
> somewhat similar to Pavel's running chunk index problem earlier in the
> year. Finding an answer is not trivial. It would apply directly to
> this problem, where the pairs are adjacent elements in the reverse
> indices vector. Any takers?
>

I've encountered a few areas where certain logic problems cannot be solved without a loop in IDL. Usually, this always points to the fact that there are certain IDL functions that (logically) insist upon scalar parameters.

-Bill B.

Subject: Re: histogram question
Posted by alt on Fri, 10 Aug 2001 11:03:57 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> wrote in message news:<3B71C2AA.7E91E5BA@astro.cornell.edu>...
> for i=0,cnt-1 do begin
> low=r[r[bad[i]] & n=r[r[bad[i]+1]]-low
> inds=indgen(n)+low
> if n_elements(list) eq 0 then list=[inds] else list=[list,inds]
> endfor
>
> now you have the list of bad indices into X in hand, to perform whatever
> punishment is necessary.
>

It seems to be some misprint or my task misunderstanding. list is
indices into r, not into x.

```
  for i=0,cnt-1 do begin
     inds = r[ r[bad[i]] : r[bad[i]+1]-1 ]
     if n_elements(list) eq 0 then list=[inds] else list=[list,inds]
  endfor
```

We needn't to check r[bad[i]] NE r[bad[i]+1] because we adding only
not empty bins.

> This brings up an interesting sub-problem though. If you have a list
> which consists of a series of pairs of indices, e.g.:
>
> [1,5,7,12,15,18]
>
> where each pair is intended to expand to the range within that pair:
>
> [1,2,3,4,5,7,8,9,10,11,12,15,16,17,18]
>
> how can you turn the former into the latter without a loop? This is
> somewhat similar to Pavel's running chunk index problem earlier in the
> year. Finding an answer is not trivial. It would apply directly to
> this problem, where the pairs are adjacent elements in the reverse
> indices vector. Any takers?

From my experience it is much faster in such cases to write a DLM
module then to rack brains on how implement something not standart
into IDL operations. It would be wonderful if IDL has build in C
compiler so we could write C code (with some limitations of course)
just inside of our IDL code. As we do in C writing ASM.

Regards,
Altyntsev Dmitriy