Subject: histogram question
Posted by Gregory Y. Prigozhin on Wed, 08 Aug 2001 20:17:51 GMT
View Forum Message <> Reply to Message

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?

Gregory

Subject: Re: histogram question
Posted by Jim Pendleton on Thu, 09 Aug 2001 22:01:57 GMT
View Forum Message <> Reply to Message

"Bill B." <billb@attinet.com> wrote in message
news:269b6343.0108090726.e32298a@posting.google.com...
>> 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.
>

Okay, here's an effort using two histograms instead of one, just to get some interest going. No visible loops, but not the sort of stuff you put in production code. With this as a head start, let's see the single-histogram approach.

```
A = [-1,3,7,12,15,18,20,20] ;The solution must work with negative and
repeating numbers
NPair = N_Elements(A)/2
MaxA = Max(A, Min = MinA)
D = MaxA - MinA
P = Lindgen(NPair)*2
H1 = Histogram(A[P], Max = MaxA, Min = MinA, R = R1)
H2 = Histogram(A[P + 1], Max = MaxA, Min = MinA, R = R2)
x1 = (R1 - MaxA)[0:D]
x2 = (R2 - MaxA)[0:D]
C = [A, ((x1 ne x2)*(Lindgen(D + 1) + MinA) > MinA) < MaxA]
C = C[Sort(C)]
C = C[Uniq(C)]
Print, C

; Loop version

B = Indgen(NPair)*2
For I = 0L, NPair - 1 Do Print, A[B[I]] + Lindgen(A[B[I] + 1]- A[B[I]] + 1)
```

Extra credit for minimal use of "[]" notation.

Jim P.

Subject: Re: Histogram question
Posted by David Fanning on Sun, 08 Aug 2004 13:40:15 GMT
View Forum Message <> Reply to Message

Michael Wallace writes:

> I have some data I need to histogram. I have two vectors, v1 and v2
> which define a two-dimensional density. Normally I could just use
> HIST_2D and and I'd be set. However, this time around I have a third
> array, v3. v3[i] corresponds to a distinct number of counts at the
> position [v1[i], v2[i]]. So, when I do the histogram, I want to use the
> value found in v3 rather than just simply calculating a density based
> only on occurrences of v1 and v2 pairs.
>
> For example, let's say that I have...
> v1 = [0, 1, 0, 2, 0, 2, 2, 1, 0]
> v2 = [1, 1, 2, 2, 0, 1, 2, 0, 0]

> v3 = [3, 0, 2, 0, 1, 1, 4, 2, 1]
>
> Doing a HIST_2D against v1 and v2 should yield something like...
> [[2, 1, 0],
> [1, 1, 1],
> [1, 0, 2]]
>
> But what I really want would use the counts in v3 instead of
> incrementing by 1 for each occurrence of [v1[i], v2[i]]...
>
> [[2, 2, 0],
> [3, 0, 1],
> [2, 0, 4]]
>
> Anyone know of an efficient way to do this? I figure there's some trick
> you can do with histogram to achieve this effect, but I am no where near
> the histogram guru like others on this list.

I don't know if this is the most "efficient" way,
but the idea is that you have to "replicate" the
numbers in v1 and v2 by the number of counts in v3.

For a quick and dirty method, I used this:

```
 ;************************************************************ ******
v1 = [0, 1, 0, 2, 0, 2, 2, 1, 0]
v2 = [1, 1, 2, 2, 0, 1, 2, 0, 0]
v3 = [3, 0, 2, 0, 1, 1, 4, 2, 1]

; Replicate each index by the number of counts.

v1_expand = Ptr_New(/Allocate_Heap)
v2_expand = Ptr_New(/Allocate_Heap)

*v1_expand = [Replicate(v1[0], v3[0])]
*v2_expand = [Replicate(v2[0], v3[0])]
FOR j=1,N_Elements(v3)-1 DO BEGIN
  IF v3[j] NE 0 THEN *v1_expand = [*v1_expand, Replicate(v1[j], v3[j])]
  IF v3[j] NE 0 THEN *v2_expand = [*v2_expand, Replicate(v2[j], v3[j])]
ENDFOR

final = Hist_2D(*v1_expand, *v2_expand)
Ptr_Free, v1_expand, v2_expand
Print, final
 ;************************************************************ ******
```

Which gave me the answer:

```
       2       2       0
       3       0       1
       2       0       4
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Histogram question
Posted by Michael Wallace on Sun, 08 Aug 2004 18:28:59 GMT
View Forum Message <> Reply to Message

```
> I don't know if this is the most "efficient" way,
> but the idea is that you have to "replicate" the
> numbers in v1 and v2 by the number of counts in v3.
>
> For a quick and dirty method, I used this:
>
> ;********************************************************** ******
> v1 = [0, 1, 0, 2, 0, 2, 2, 1, 0]
> v2 = [1, 1, 2, 2, 0, 1, 2, 0, 0]
> v3 = [3, 0, 2, 0, 1, 1, 4, 2, 1]
>
> ; Replicate each index by the number of counts.
>
> v1_expand = Ptr_New(/Allocate_Heap)
> v2_expand = Ptr_New(/Allocate_Heap)
>
> *v1_expand = [Replicate(v1[0], v3[0])]
> *v2_expand = [Replicate(v2[0], v3[0])]
> FOR j=1,N_Elements(v3)-1 DO BEGIN
>    IF v3[j] NE 0 THEN *v1_expand = [*v1_expand, Replicate(v1[j], v3[j])]
>    IF v3[j] NE 0 THEN *v2_expand = [*v2_expand, Replicate(v2[j], v3[j])]
> ENDFOR
>
> final = Hist_2D(*v1_expand, *v2_expand)
> Ptr_Free, v1_expand, v2_expand
> Print, final
> ;********************************************************** ******
>
> Which gave me the answer:
>
```

> 2 2 0
> 3 0 1
> 2 0 4
>

Thanks, David. While this does work, using replicate could potentially create some monster-sized arrays. I failed to mention before that in a normal case, the values in the v3 vector will be on the order of thousands... and sometimes 10s of thousands. That's going to be a lot of numbers in memory!

I wrote this little snippet which doesn't use HIST_2D at all. It's just a simple FOR loop and adding values. Back in the good ol' days, I would have been satisfied with this, but after having read this newsgroup for a while I just have a feeling that there's a way to get rid of that FOR loop and do something totally cool with the HISTOGRAM function. So I guess my question is more academic than anything else. Anyway, here's the snippet ...

```
v1 = [0, 1, 0, 2, 0, 2, 2, 1, 0]
v2 = [1, 1, 2, 2, 0, 1, 2, 0, 0]
v3 = [3, 0, 2, 0, 1, 1, 4, 2, 1]


arr = intarr(max(v1) + 1, max(v2) + 1)
n = n_elements(v3)

for i = 0, n - 1 do begin
    arr[v1[i], v2[i]] += v3[i]
endfor


print, arr
```

-Mike

Subject: Re: Histogram question
Posted by mperrin+news on Sun, 08 Aug 2004 18:52:57 GMT
View Forum Message <> Reply to Message

Michael Wallace <mwallace.no.spam@no.spam.swri.edu.invalid> wrote:
> I wrote this little snippet which doesn't use HIST_2D at all. It's just
> a simple FOR loop and adding values. Back in the good ol' days, I would
> have been satisfied with this, but after having read this newsgroup for
> a while I just have a feeling that there's a way to get rid of that FOR

> loop and do something totally cool with the HISTOGRAM function. So I
> guess my question is more academic than anything else.

It's too bad HIST_2D doesn't have a REVERSE_INDICES keyword like HISTOGRAM does. If it did, you could do something like

```
H = hist_2d(v1,v2,reverse_indices=r)
for i=0L,n_elements(h)-1 do
 H[i] +=  total(v3[R[R[I] : R[i+1]-1]] -1 )
```

You could maybe still make this work if you don't mind unrolling your 2D arrays into 1D arrays somehow. The above code is just off the top of my head and hasn't been tested or debugged or anything like that, so no promises are made!

- Marshall

Subject: Re: Histogram question
Posted by Chris Lee on Mon, 09 Aug 2004 14:39:17 GMT
View Forum Message <> Reply to Message

> .. However, this time around I have a third
> array, v3. v3[i] corresponds to a distinct number of counts at the
> position [v1[i], v2[i]]. So, when I do the histogram, I want to use the
> value found in v3
> .. v1 = [0, 1, 0, 2, 0, 2, 2, 1, 0]
> v2 = [1, 1, 2, 2, 0, 1, 2, 0, 0]
> v3 = [3, 0, 2, 0, 1, 1, 4, 2, 1]
>
> ans=[[2, 2, 0],
> [3, 0, 1],
> [2, 0, 4]]
> Anyone know of an efficient way to do this? I figure there's some trick
> you can do with histogram to achieve this effect, but I am no where near
> the histogram guru like others on this list. -Mike

There are some things that HISTOGRAM can't do (no, really). TOTAL and WHERE can help you a bit. I can get the loop down to the total number of bins in the histogram (in this case there really isn't any speed inprovement, hopefully you have more elements than bins in the real data?)

If your data has lots of zeros in v3 then this method gets more efficient than the 'count everything' approach because the zeros are silently ignored. (If your resulting histogram is sparse another, more tedious optimisation, can be made by doing a HIST_2D(v1,v2) before hand)

```
v1 = [0, 1, 0, 2, 0, 2, 2, 1, 0]
v2 = [1, 1, 2, 2, 0, 1, 2, 0, 0]
v3 = [3, 0, 2, 0, 1, 1, 4, 2, 1]

n=n_elements(v1)
;build an index array
index=fltarr(n)
index=v2*(max(v1)+1)+v1
;m=the number of bins in the 2d histogram
m=max(index)+1
;tot=the 2d histogram
tot=fltarr(m)

for i=0, m-1 do begin & $
  w=where(index eq i,c)  & $
  if(c gt 0) then tot[i]=tot[i]+total(v3[w]) & $
endfor

tot=reform(tot, max(v1)+1, max(v2)+1)
;
;
```

Chris.

Subject: Re: Histogram question
Posted by JD Smith on Tue, 10 Aug 2004 18:43:40 GMT
View Forum Message <> Reply to Message

On Sun, 08 Aug 2004 18:52:57 +0000, Marshall Perrin wrote:

> Michael Wallace <mwallace.no.spam@no.spam.swri.edu.invalid> wrote:
>> I wrote this little snippet which doesn't use HIST_2D at all. It's just
>> a simple FOR loop and adding values. Back in the good ol' days, I would
>> have been satisfied with this, but after having read this newsgroup for
>> a while I just have a feeling that there's a way to get rid of that FOR
>> loop and do something totally cool with the HISTOGRAM function. So I
>> guess my question is more academic than anything else.
>
> It's too bad HIST_2D doesn't have a REVERSE_INDICES keyword like HISTOGRAM
> does. If it did, you could do something like
>
> H = hist_2d(v1,v2,reverse_indices=r)
> for i=0L,n_elements(h)-1 do
> H[i] += total(v3[R[R[I] : R[i+1]-1]] -1)
>
>
> You could maybe still make this work if you don't mind unrolling your 2D

> arrays into 1D arrays somehow.

That's how HIST_2D does it's magic. The HIST_ND program I wrote also
gives you reverse indices (find it on David's site:
http://www.dfanning.com/programs/hist_nd.pro).

JD

Subject: Re: Histogram question
Posted by David Fanning on Tue, 10 Aug 2004 19:26:07 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> That's how HIST_2D does it's magic. The HIST_ND program I wrote also
> gives you reverse indices (find it on David's site:
> http://www.dfanning.com/programs/hist_nd.pro).

Just put the latest on the site a minute ago.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
