Subject: Re: Speed-up of code
Posted by David Fanning on Tue, 25 Aug 2009 15:01:47 GMT
View Forum Message <> Reply to Message

Philip Elson writes:

> My assumption is that the HISTOGRAM function will be helpful, but
> having spent quite some time on this I am beginning to think that it
> cannot be done - though I would love to be proved wrong by any
> histogram guru out there.

Just a note for any new members of the IDL newsgroup
out there. THIS is the way you ask a question that
gets results! :-)

Cheers,

David

--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Speed-up of code
Posted by wlandsman on Tue, 25 Aug 2009 15:36:15 GMT
View Forum Message <> Reply to Message

On Aug 25, 10:16 am, Philip Elson <philipel...@googlemail.com> wrote:

> My assumption is that the HISTOGRAM function will be helpful, but
> having spent quite some time on this I am beginning to think that it
> cannot be done - though I would love to be proved wrong by any
> histogram guru out there.

I think you want to read the "drizzling" article on David's Web page
(http://www.dfanning.com/code_tips/drizzling.html). (Your question
differs only in that in you want to average the values rather than sum
them).

I think "Histogram plus cumulative total" usually had the winning
time:

```
h = HISTOGRAM(day, REVERSE_INDICES=ri)
nh = N_ELEMENTS(h)
sortData = value[ri[nh+1:*]]
```

```
totSortData = [0., TOTAL(sortData, /CUMULATIVE)]
vec8 = totSortData[ri[1:nh]-nh-1] - $
      totSortData[ri[0:nh-1]-nh-1]
print,vec8/h  ;May need to check for zero values first
```

--Wayne

Subject: Re: Speed-up of code
Posted by Craig Markwardt on Tue, 25 Aug 2009 15:42:29 GMT
View Forum Message <> Reply to Message

On Aug 25, 10:16 am, Philip Elson <philipel...@googlemail.com> wrote:
> Dear All,
>
> I have a question relating to the optimization of some code which
> averages an array based on the values in another array.
> Its much easier to explain in an example:
>
> day   = [1, 1, 2, 3, 3, 3, 3]
> value = [2, 4, 5, 2, 3, 2, 1]
>
> Which should return, depending on which is easier, either
> avg   = [3, 5, 2]
> or
> avg   = [3, 3, 5, 2, 2, 2, 2]
>
> This is fairly straightforward using a for loop, but how to do it in
> the IDL way?
>
> You can see two examples of the basic code below:
>

```
> ; ================================
> ;                FIRST EXAMPLE
> ; ================================
> unique = uniq(day)
> avg = intarr(n_elements(unique))
> FOR i=0, n_elements(unique) -1 DO BEGIN
>   res = WHERE(day EQ day[unique[i]], count)
>   if count GT 0 THEN avg[i] = total(value[res],/DOUBLE) / count
> ENDFOR
> print, avg
>
> ; ================================
> ;                SECOND EXAMPLE
> ; ================================
> h = histogram(day, REVERSE_INDICES=ri)
> avg = h*0
```

> FOR i=0, n_elements(h)-1 DO BEGIN
> data_inds = ri[ri[i]:ri[i+1]-1]
> avg[i] = total(value[data_inds],/DOUBLE) / h[i]
> ENDFOR
> print, avg
>
> At this stage I open the floor; I essentially want to achieve the
> results as above without the need for the for loop.
>
> My assumption is that the HISTOGRAM function will be helpful, but
> having spent quite some time on this I am beginning to think that it
> cannot be done - though I would love to be proved wrong by any
> histogram guru out there.

Those are the techniques I would have tried! Be careful: in your second example, you don't handle the case where the histogram bin h[i] is empty. You just need an "if h[i] GT 0" test there.

Craig

Subject: Re: Speed-up of code
Posted by philipelson on Tue, 25 Aug 2009 16:12:48 GMT
View Forum Message <> Reply to Message

> h = HISTOGRAM(day, REVERSE_INDICES=ri)
> nh = N_ELEMENTS(h)
> sortData = value[ri[nh+1:*]]
> totSortData = [0., TOTAL(sortData, /CUMULATIVE)]
> vec8 = totSortData[ri[1:nh]-nh-1] - $
> totSortData[ri[0:nh-1]-nh-1]
> print,vec8/h ;May need to check for zero values first

Aha! Thank you all very much!

If I was to extend this further, any suggestions on the obtaining the maximum rather than the average of each bin?

This would indeed provide some nice examples of gathering basic statistics using the histogram function!

Many Thanks,

Philip

Subject: Re: Speed-up of code
Posted by philipelson on Tue, 25 Aug 2009 16:14:13 GMT
View Forum Message <> Reply to Message

P.S. I promise not to move the goalposts again! :-)

Subject: Re: Speed-up of code
Posted by philipelson on Tue, 25 Aug 2009 16:18:56 GMT
View Forum Message <> Reply to Message

> Those are the techniques I would have tried! Be careful: in your
> second example, you don't handle the case where the histogram bin h[i]
> is empty. You just need an "if h[i] GT 0" test there.

Wayne,

To fix the note you have added in your code

> print,vec8/h ;May need to check for zero values first

I added the line:

print,vec8/(h>1)

Which does the trick nicely.

Regards,

Philip

Subject: Re: Speed-up of code
Posted by JDS on Tue, 25 Aug 2009 22:30:48 GMT
View Forum Message <> Reply to Message

On Aug 25, 12:12 pm, Philip Elson <philipel...@googlemail.com> wrote:
>> h = HISTOGRAM(day, REVERSE_INDICES=ri)
>> nh = N_ELEMENTS(h)
>> sortData = value[ri[nh+1:*]]
>> totSortData = [0., TOTAL(sortData, /CUMULATIVE)]
>> vec8 = totSortData[ri[1:nh]-nh-1] - $
>> totSortData[ri[0:nh-1]-nh-1]
>> print,vec8/h ;May need to check for zero values first
>
> Aha! Thank you all very much!
>
> If I was to extend this further, any suggestions on the obtaining the

> maximum rather than the average of each bin?
>
> This would indeed provide some nice examples of gathering basic
> statistics using the histogram function!

The "dual histogram" method is useful for this. I'd use it like this:

```
h=histogram(day,REVERSE_INDICES=ri)
mx=fltarr(n_elements(h))
h2=histogram(h,OMIN=om,REVERSE_INDICES=ri2)
for k=long(om eq 0),n_elements(h2)-1 do begin
  if h2[k] eq 0 then continue
  bins=ri2[ri2[k]:ri2[k+1]-1]  ;bins of the histogram with this
repeat count

  if k+om eq 1 then begin
    mx[bins]=value[ri[ri[bins]]]
  endif else begin
    targ=[h2[k],k+om]
    points=ri[ri[rebin(bins,targ)]+rebin(lindgen(1,k+om),targ)]
    mx[bins]=max(value[points],DIMENSION=2)
  endelse
endfor
```

You'll note this does use a loop, but it's a loop over the number of different bin occupation counts, i.e. a very small loop typically, even with very large inputs. Please note that this is really only useful compared to your 2nd example above only if you need speed at the expense of clarity. You might want to set the array to NaN originally, to notice any "missing" days in this example. Beware that the cumulative total method of binning suffers greater round-off error than the other methods, so you might be advised to use it with /DOUBLE.

JD
