Subject: Histogram Hot-shots Required
Posted by davidf on Thu, 15 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Ok you histogram cowboys. Let's see what ya got!

I don't know if it's early on-set Alzheimers, or
I'm just pressing with a long soccer weekend coming
up, or I just haven't had enough beers yet, but
this one is stumping me. I thought I'd give you
folks a chance to see if you can explain something
simply enough that even I can understand it. :-)

Here's what I've got. I have an array (call it an
image) in one window, and a plot of the array's
histogram in another window. I want to plot
the pixel density along the vertical Y axis and
the image pixel value along the X axis (rather
than, say, the bin number). So my code looks
roughly like this:

```
  histdata = info.image->Histogram()
  imgRange = info.image->GetImageRange()
  bins = Findgen(N_Elements(histdata))
  bins = Scale_Vector(bins, Min(imgRange), Max(imgRange))
  xrange = [Min(bins), Max(bins)]
  Plot, bins, histdata, YTitle='Pixel Density', $
     XTitle='Bin Value' , Title=title, $
     background=charcoal, Color=green, /NoData, $
     XRange=xrange, XStyle=1
  OPlot, bins, histdata, Color=yellow
```

Yeah, yeah, it's an image object, but the Histogram
method is just basically returning the results of
the HISTOGRAM function. The method does other things
like calculate the binsize based on the type of data
I have in the image window, etc. The Scale_Vector
function just scales the bin values so they cover
the entire range of array values.

Now, here is the neat part. As I move my cursor
in the image window, I obtain the image value under
the cursor. I want to plot this value on my histogram
plot as a vertical line. No problem. I just
restore the proper plotting system variables and use
PlotS like this:

```
  PlotS, [value, value], !Y.CRange, Color=red
```

This works great. Because of the way I scaled the
bins and set up the X axis I can get the line on
the histogram plot in exactly the location I want
it in.

But here is what I don't get. What I want now
(well, this is really a matter of a what my *client*
wants now) is to print out on the histogram plot not
just the value of the image at the cursor location,
but the pixel density at that location. In other words,
this pixel belongs to a particular bin. I want to print
out the total number of other pixels that also belong
in that bin.

What I can't figure out tonight is how to find out
what bin that pixel is in, given that I know the pixel
value. (Even as I write this sentence I have the sense
that this is a trivial exercise, but I'm afraid it is
not yielding the shear number of hours I have spent
on it. At least not for me.)

I've tried a number of things, most of which I'm too
embarrassed to mention. My most promising result looks
like this:

```
  binNumbers = Where(value LT bins)
  binNum = binNumbers[0] - 1
  XYOuts, 0.975, 0.05, Align=1.0, 'Pixel Density: ' + $
    StrTrim(histData[binNum], 2), /Normal, Color=yellow
```

This *almost* works, but a close look at the value and the
graph shows some discrepancy. The vertical value line
will cross the graph at, say, 75 and the pixel density will
be calculated at 92. I've tried changing the bin size, and
hence the number of bins in the resulting histogram, but
this doesn't seem to help.

In any case, I'm fresh out of ideas as well as beer. So
I thought I'd turn it over to you. Any ideas will be
gratefully accepted. I'm sure it has something to do
with that Reverse_Indices keyword, but whatever it is
escapes me. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Histogram Hot-shots Required
Posted by eddie haskell on Fri, 16 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> What I can't figure out tonight is how to find out
> what bin that pixel is in, given that I know the pixel
> value.

David,

I believe reverse_indices are the way to go. First the short answer:

given:
A - the array containing the data (your image)
V - the data (pixel) value at the chosen location
R - the reverse_indices returned from the histogram function
    (and I don't care what you chose as a bin size, that is incorporated
     into R)

The number of the bin in which the data value resides can be calculated
as:

bin=(where(R ge ((where(((where(A eq V))[0]) eq
R[R[0]:*]))[0]+R[0])))[0]

and the number of elements in that bin is then:

num=R[bin+1]-R[bin]

Now the long answer:

reverse_indices gives you the number of elements in each bin and tells
you in which bin each element of your original array falls.

For example:
A=[1,2,3,4,3,2,2]
h=histogram(A,bin=2,reverse=R) ;note: binsize does not matter to this
 function, I chose 2 to make R smaller.
  now R = [3, 7, 10, 0, 1, 5, 6, 2, 3, 4]

What R is telling you is that the subscripts of the items in A in the
first bin are the ones found in R[3:7] or A[0,1,5,6] which are the
positions of the 1's and 2's in A. The second bin is R[7:10] or
A[2,3,4], the 3's and 4.

Choose a value that does exist in the array A: e.g., V = 3
First determine a location in A where V exists, for the sake of ease we
will choose the first occurance of V

```
w1=(where(A eq V))[0]   ;here w1 = 2
```

Since we know V has to exist in A we don't have to check for where
returning -1

Then we find the location of that position in the vector R

```
w2=(where(w1 eq R[R[0]:*]))[0]+R[0]    ;here w2 = 7
```

R[0] happens to be the index of the first location of the portion of R
where the separate array subscripts are stored. Adding R[0] to the
where() returns the actual value. Again, we know the value exists so no
need to check for a -1

The first part of R lists the indices in R of where the subscripts of A
are found in each individual bin. Just look for the first value that is
greater than or equal to w2.

```
bin=(where(r ge w2))[0]   ;here bin = 1 or the value of 3 is found in
the
 2nd bin.
```

The long line at the top condenses these steps into one confusing line.
As before, finding the number of elements in that bin is
straightforward.

```
num=R[bin+1]-R[bin]
```

I hope this helps you. As I noted before, odds are either this solution
will not work for you or a better solution will soon present itself.

Cheers,
eddie

----- ---- --- --- ---- --- -- --- --- -- -- - - - -
|\   A G Edward Haskell
|\   Center for Coastal Physical Oceanography
|\   Old Dominion University, Norfolk VA 23529

```
|\     Voice 757.683.4816    Fax 757.683.5550
|\         e-mail  haskell*ccpo.odu.edu
----- ---- --- ----  --- --- ---  --- -- --  -- - -  -   -
```

Subject: There is NO TRUTH! Re: Histogram Hot-shots Required
Posted by davidf on Mon, 19 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Folks,

More evidence that the apocalypse is upon us....

Here are five different methods for calculating the
histogram bin number from a value:

```
       ; Liam's method 1:

   binNum1 = FLOOR((value - Min(array)) / binsize)

     ; Liam's method 2:

   binNum2 = ROUND((value - Min(array)) / binsize)

       ; Liam's method 3:

   binNum3 = CEIL((value - Min(array)) / binsize)

      ; Eddie's method:

   binNum4 =(where(R ge ((where(((where(array eq value))[0]) eq $
      R[R[0]:*]))[0]+R[0])))[0]

       ; My brute-force, no-excuses method:

    FOR j=0,N_Elements(bins)-2 DO BEGIN
      IF value GT bins[j] and value LT bins[j+1] THEN BEGIN
         binNum5 = j
         goto, getout
      ENDIF
     ENDFOR
     getout:
```

Here is typical output of this print statement:

```
   Print, binNum1, binNum2, binNum3, binNum4, binNum5

          34         35         35         35      34
```

```
    11      12      12      12    11
    11      11      12      12    11
    13      13      14      14    13
     2       3       3       3     2
    10      11      11      11    10
    12      12      13      13    12
    28      28      29      29    28
    26      27      27      27    26
    33      34      34      34    33
    34      35      35      35    34
    32      33      33      33    32
```

Notice that method 1 and method 5 always produce the same
number. Whew! It may be brute-force, but I can explain it
and I am SURE it is right!

But, woe of woes, here is the graphical output for the last
value:

  http://www.dfanning.com/images/histogram.jpg

Not even close. :-(

The best graphical values are STILL method 3.

HELP!!!!!

Cheers,

David

P.S. Let's just say my confidence in rational thought is slipping...

Of course, I'm listening to my Bob Marley CD, so that may have
something to do with it :-)

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Histogram Hot-shots Required
Posted by davidf on Mon, 19 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Eddie Haskell (haskell@see.signature.edu) writes:

> The number of the bin in which the data value resides can be calculated
> as:
>
> bin=(where(R ge ((where(((where(A eq V))[0]) eq
> R[R[0]:*]))[0]+R[0])))[0]
>
> and the number of elements in that bin is then:
>
> num=R[bin+1]-R[bin]

Just because I know you folks like a little fun...

I don't really understand this little equation of Eddie's,
but I thought I would implement it anyway and compare it
to the example Liam and other's sent me. Here is the way
I implemented my code. The Histogram method is returning,
via the keywords, the original array that the histogram is
performed on, the bin size, and the reverse indices. Value,
you recall, is the value I have and I want to know which bin
this value is in.

```
  histdata = info.image->Histogram(Data=array, Title=title, $
     Binsize=binsize, Reverse_Indices=r)

  bins = (Findgen(N_Elements(histdata)) * binsize) + Min(array)

     ; Liam's method:

  binNum = Round((value - Min(array)) / binsize)

     ; Eddie's method:

  binNum =(where(R ge ((where(((where(array eq value))[0]) eq $
     R[R[0]:*]))[0]+R[0])))[0]

  pixelDensity = histdata[binNum]
```

Since the bin number must be an integer, I experimented
with taking the FLOOR, CEIL, and ROUND of the calculated
value using Liam's method.

Here is what I learned. Most of the time, the two methods
are pretty good at finding the same bin number, but not
always. If I take the FLOOR of Liam's calculated value, the
bin number is pretty consistently one LESS than Eddie's
number. If I take the CEIL of Liam's value, the values

are mostly the same, but not always. If I take the
ROUND of the value, I see a similar pattern: mostly the
same, but not always.

Which number is most consistent with the graph of the
Histogram plot, do you think?

```
   Plot, bins, histData, PSym=10
   PLOTS, [value, value], !Y.CRange
```

Well, this surprised me, but it is Liam's number with
the ROUNDing.

Now...what do you make of that? Which do you think is
really more accurate?

And, then, tonight I was reading the release notes for IDL
5.2.1 and I found this:

   HISTOGRAM Function Error with BINSIZE Set Fixed:

      The HISTOGRAM function error resulting when the BINSIZE
      keyword is set has been fixed in this release.

Could *this* be the problem? What *was* that problem, anyway?

I'm still confused. I don't like code with fudge factors and
algorithms I arrive at by trial and error. :-(

Let's just say I'm awaiting further insight...

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Histogram Hot-shots Required
Posted by Liam Gumley on Tue, 20 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:

> Liam Gumley <Liam.Gumley@ssec.wisc.edu> wrote in message
> news:378F33E4.984726C5@ssec.wisc.edu...
>> Assuming I know the minimum and maximum values (the range) used in
>> creating the histogram, the histogram binsize, and the number of bins,
>> the zero-based bin index is given by
>>
>> bin_index = long(float(pixel_value - histogram_min_value) /
>> float(binsize))
>>
>> and then to protect against pixel values LT histogram minimum value, or
>> GE histogram maximum value
>>
>> bin_index = (bin_index > 0L) < (number_of_bins - 1L)
>>
>
> I agree with the first part, but HISTOGRAM ignores values outside the
> specified range, so you would want to check the index and throw away if LT 0
> or GE "num_bins"

which is done by the the statement

bin_index = (bin_index > 0L) < (number_of_bins - 1L)

Cheers,
Liam.

--
Liam E. Gumley
Space Science and Engineering Center, UW-Madison
http://cimss.ssec.wisc.edu/~gumley

Subject: Re: Histogram Hot-shots Required
Posted by Martin Downing on Tue, 20 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Liam Gumley <Liam.Gumley@ssec.wisc.edu> wrote in message
news:378F33E4.984726C5@ssec.wisc.edu...
> Assuming I know the minimum and maximum values (the range) used in
> creating the histogram, the histogram binsize, and the number of bins,
> the zero-based bin index is given by
>
> bin_index = long(float(pixel_value - histogram_min_value) /
> float(binsize))
>
> and then to protect against pixel values LT histogram minimum value, or
> GE histogram maximum value
>

> bin_index = (bin_index > 0L) < (number_of_bins - 1L)
>

I agree with the first part, but HISTOGRAM ignores values outside the
specified range, so you would want to check the index and throw away if LT 0
or GE "num_bins"

Martin

Subject: Re: There is NO TRUTH! Re: Histogram Hot-shots Required
Posted by davidf on Wed, 21 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Gabriel Rodriguez (gabriel@gbt.tfo.upm.es) writes:

> Maybe the problem is how IDL makes the 'histogram' plot. According to the
> manual setting the PSYM=10 is:
>
> "Histogram mode. Horizontal and vertical lines connect the plotted
> points, as opposed to the normal method of connecting points with
> straight lines."
>
> But it is not clear for me if the point is in the middle of each 'step'
> in the histogram, or to the left or the right of each step.

This is the problem. The point is in the middle of each step.
But the bin goes from the start of one step to the start of
the next.

> From what I see in your plot: if you move the vertical line one half
> 'binsize' or 'step' to the left you might be getting the right value?

Yes, exactly. But if I move the vertical line one half bin size,
then I don't have the right *value*, and that, after all, is what
I am after here. :-) The vertical line is in the right place.
It is the histogram plot that is screwy. Although I realize
no one believes me, I think the ONLY solution is to draw your
own histograms. :-)

> One posible way to see it is displaying a symbol where the exact value
> is.

Exactly. This is, in fact, how I eventually solved the problem.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: There is NO TRUTH! Re: Histogram Hot-shots Required
Posted by gabriel rodriguez ibe on Wed, 21 Jul 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Maybe the problem is how IDL makes the 'histogram' plot. According to the
manual setting the PSYM=10 is:

"Histogram mode. Horizontal and vertical lines connect the plotted
points, as opposed to the normal method of connecting points with
straight lines."

But it is not clear for me if the point is in the middle of each 'step'
in the histogram, or to the left or the right of each step.

From what I see in your plot: if you move the vertical line one half
'binsize' or 'step' to the left you might be getting the right value?

One posible way to see it is displaying a symbol where the exact value
is.