Subject: Re: Automatic Binsize Calculations
Posted by manodeep@gmail.com on Mon, 30 May 2011 18:26:25 GMT
View Forum Message <> Reply to Message

```
On May 30, 8:47 am, Craig Markwardt <craig.markwa...@gmail.com> wrote:
>  On May 29, 12:42 pm, David Fanning <n...@idlcoyote.com> wrote:
>
>
>
>
>
>
>
>
>
>>  Gianguido Cianci writes:
>>>  Here's what I came up with, using sshist_2d.pro
>>>  (http://tinyurl.com/3on7bzx) that automagically finds bin size:
>
>>  I don't have a television, so while I listened to Djokovic
>>  defeat Gasquet on the French Open Radio I was fooling
>>  around using the 1D version of sshist to calculate
>>  a default bin size for cgHistoplot. What I discovered
>>  is that I get completely different results depending
>>  on the data type of the input data!
>
>>  I modified sshist a bit to get the bin size out of it
>>  as a keyword:
>
>>  ; Author: Shigenobu Hirose at JAMSTEC
>>  ; based on original paper
>>  ; Shimazaki and Shinomoto, Neural Computation 19, 1503-1527, 2007
>>  ; http://toyoizumilab.brain.riken.jp/hideaki/res/histogram.htm l
>>  ;
>>  function sshist, data, x=x, cost=cost, nbin=nbin, binsize=binsize
>
>>    COMPILE_OPT idl2
>
>>    nbin_min = 2
>>    nbin_max = 200
>
>>    ntrial = nbin_max - nbin_min + 1
>
>>    nbin  = INDGEN(ntrial) + nbin_min
>
>>    delta = FLTARR(ntrial)
>>    cost  = FLTARR(ntrial)
>
```

```
>>    for n = 0, ntrial-1  do begin
>>       delta[n] = (MAX(data) - MIN(data)) / (nbin[n] - 1)
>
>>       k = HISTOGRAM(data, nbins=nbin[n])
>
>>       kmean = MEAN(k)
>>       kvari = MEAN((k - kmean)^2)
>>       cost[n] = (2. * kmean - kvari) / delta[n]^2
>>    endfor
>
>>    n = (WHERE(cost eq MIN(cost)))[0]
>>    k = HISTOGRAM(data, nbins=nbin[n], locations=x, reverse_indices=ri)
>
>>    if arg_present(binsize) then binsize = delta[n]
>>    return, k
>
>>  end
>
>>  But, look at this:
>
>>  IDL> void = sshist(cgdemodata(21), binsize=bs) & print, bs
>>        9.00000
>>  IDL> void = sshist(fix(cgdemodata(21)), binsize=bs) & print, bs
>>        1.00000
>>  IDL> void = sshist(long(cgdemodata(21)), binsize=bs) & print, bs
>>        1.00000
>>  IDL> void = sshist(float(cgdemodata(21)), binsize=bs) & print, bs
>>        1.33684
>
>>  I have NO idea why this is occurring. :-(
>
>  I think you have more than one thing going on, which is making things
>  more confusing than otherwise.
>
>  First, it looks like there is a serious bug in HISTOGRAM, which
>  produces *negative* counts for byte data.  Check this out:
>  IDL> print, histogram(cgdemodata(21), nbins=nbin[n])
>        13591       43618      108702       55359       37621
>  15767
>         9343  -975994564
>  Huh?? *Negative* 1 billion?  This bug exists in IDL7, so it's been
>  around for a while.  I can't believe this hasn't showed up before!
>
```

It gets even more weird:

```
IDL> print,!version
```

```
{ x86_64 linux unix linux 8.0 Jun 18 2010      64      64}

IDL> xx = cgdemodata(21)
IDL> print,total(histogram(xx,nbin=16b,min=0b,max=255b),/pres)
  1798093803
IDL>  print,total(histogram(cgdemodata(21),nbin=16b,min=0b,max=255 b),/
pres)
 -2145213021

IDL>  print,total(histogram(cgdemodata(21),nbin=16b,min=0b,max=255 b),/
pres)
 -2145229853
```

And the last one output changes arbitrarily. So now the result is dependent on whether a named variable is passed into histogram or not. Now if I just change the min keyword for histogram, I get:

```
IDL>  print,total(histogram(cgdemodata(21),nbin=16b,min=1b,max=255 b),/
pres)
      168048
IDL> print,total(histogram(xx,nbin=16b,min=1b,max=255b),/pres)
      168048
IDL> print,n_elements(xx)
      168048
```

So that looks good, i.e., no negative histogram counts. Histogram (with min=0b) still produces negative counts - so the bug is intrinsic to the way histogram is handling byte data. Setting some of the xx values to 0b doesn't change the billion particle count in the final bin. I guess what's happening is that the binwidth is > 1b, therefore the last bin actually also contains numbers that are also between [0b-15b].

```
IDL> print,total(histogram(xx,nbin=256,min=0b,max=255b),/pres)
      168048
```

Voila. The binsize now means no wrapping and there are no issues with histogram. Note that you can not set nbins > 256, since binsize will become 0b (and histogram will complain about illegal binsize).

Cheers,
Manodeep