Subject: Re: Automatic Binsize Calculations
Posted by manodeep@gmail.com on Sun, 29 May 2011 18:20:33 GMT
View Forum Message <> Reply to Message

On May 29, 11:42 am, 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. :-(
>

If I set the "x" keyword to sshist, I see that the range returned is:
(note, cgdemodata(21) returns a [432,389] byte array ranging between 1
and 255 for me)

byte : 0-255 [bs = 84 and not 9 like David has]
int : 1-147 [bs = 1]
long : 1-147 [bs = 1]
float: 1-255 [bs = 1.33]

There must be a data-type mismatch going on somewhere. Only the float
calculation returns the histogram for the actual data range.
If I change the delta[n] line in sshist to

delta[n] = (max(data) - min(data))/(nbin[n] - 1.0)

i.e., force the calculation to be in floating point, then the int/long
types also return the range 1-255 (with a binsize of 2.0). The byte
calculation still has the same range but bs changes to 84.66. Not
entirely sure I understand what is going on..

Cheers,
Manodeep

Subject: Re: Automatic Binsize Calculations

On Sun, 29 May 2011, David Fanning 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. :-(
>
> Cheers,
>
> David

My result is worse:

```
IDL> print, !version
{ x86_64 linux unix linux 8.1 Mar  9 2011      64      64}
IDL> void = sshist(cgdemodata(21), binsize=bs) & print, bs
% Compiled module: CGDEMODATA.
% Loaded DLM: JPEG.
% Compiled module: SSHIST.
% Compiled module: MEAN.
% Compiled module: MOMENT.
% Array dimensions must be greater than 0.
% Execution halted at: SSHIST            26    .../sshist.pro
%                    $MAIN$
IDL>
```

Removing 'reverse_indices=ri' from histogram:

```
IDL> void = sshist(cgdemodata(21), binsize=bs) & print, bs
      21.0000
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
```

regards,
Lajos

Subject: Re: Automatic Binsize Calculations
Posted by Craig Markwardt on Mon, 30 May 2011 13:47:39 GMT
View Forum Message <> Reply to Message

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!

But you also need to be careful about float vs. integer. Your line,

```
    delta[n] = (MAX(data) - MIN(data)) / (nbin[n] - 1)
```

doesn't always work right if data is an integer type due to rounding issues. I changed that to,

```
    delta[n] = (MAX(data) - MIN(data) + 0.) / (nbin[n] - 1)
```

I also worked around the bug in HISTOGRAM inside the loop by using
this bit of extra code:
    ;; Work around an apparent bug in HISTOGRAM for BYTE
data,
    ;; which can produce corrupt data in the final
bin.
    k = HISTOGRAM(data, nbins=nbin[n]+1)
    k = k[0:nbin[n]-1]

And now more stable numbers come out of your function.

Craig

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

Subject: Re: Automatic Binsize Calculations
Posted by chris_torrence@NOSPAM on Fri, 29 Jul 2011 20:44:30 GMT
View Forum Message <> Reply to Message

On Monday, May 30, 2011 7:47:39 AM UTC-6, Craig Markwardt wrote:
>
> 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!
>

Hi all,

Just FYI, this was indeed a bug in HISTOGRAM with byte data, that has been there since IDL
-975994564.
It is fixed for IDL 8.2.

Cheers,
Chris
ITTVIS
