

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Take the 2-minute tour x

Histogram using gnuplot?



I know how to create a histogram (just use "with boxes") in gnuplot if my .dat file already has properly binned data. Is there a way to take a list of numbers and have gnuplot provide a histogram based on ranges and bin sizes the user provides?

gnuplot histogram binning

asked Mar 18 '10 at 17:10

 **mary**
742 3 9 10

2 If you don't get an answer there are other tools which are meant to do such things. I use Root (root.cern.ch) many others around here use R, and there are at least a few other options. – [dmckee](#) Mar 18 '10 at 17:15

41,000 views and only 41 upvotes for the question and 55 for the answer... – [a different ben](#) Nov 9 '12 at 5:00

what is bin and what is binned? – [Wakan Tanka](#) May 19 at 13:54

oh I got it answers.yahoo.com/question/index?qid=20101219231233AASddbx – [Wakan Tanka](#) May 19 at 14:02

9 Answers

yes, and its quick and simple though very hidden:

```
binwidth=5
bin(x,width)=width*floor(x/width)

plot 'datafile' using (bin($1,binwidth)):(1.0) smooth freq with boxes
```


check out `help smooth freq` to see why the above makes a histogram

to deal with ranges just set the xrange variable.

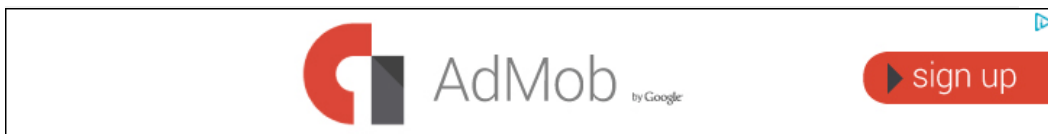
edited Feb 8 '14 at 23:38

answered Mar 29 '10 at 14:52

 **Matt Ball**
189k 37 337 419

 **Born2Smile**
1,708 1 8 8

4 I think @ChrisW's answer below brings an important point to notice for anyone who wants to make a Histogram in Gnuplot. – [Abhinav](#) Oct 26 '13 at 2:49




I have a couple corrections/additions to Born2Smile's very useful answer:

1. Empty bins caused the box for the adjacent bin to incorrectly extend into its space; avoid this using `set boxwidth binwidth`
2. In Born2Smile's version, bins are rendered as centered on their lower bound. Strictly they ought to extend from the lower bound to the upper bound. This can be corrected by modifying the `bin` function: `bin(x,width)=width*floor(x/width) + binwidth/2.0`

edited Jan 7 '13 at 12:25

 **mgilson**
114k 12 161 255

answered Aug 28 '10 at 22:03

 **mas90**
681 5 2

-
- 10 Actually that second part should be `bin(x,width)=width*floor(x/width) + binwidth/2.0` (floating point calculations) – [PiPeep](#) Aug 29 '10 at 18:33
-
- 4 You mean `bin(x,width)=width*floor(x/width) + width/2.0` . If we are passing `width` as an argument, then use it. :-) – [Mitar](#) May 15 '13 at 23:49
-
- 2 @mgilson i think ChrisW's answer brings an important correction to this answer. – [Abhinav](#) Oct 26 '13 at 3:07
-

Be very careful: all of the answers on this page are implicitly taking the decision of where the binning starts - the left-hand edge of the left-most bin, if you like - out of the user's hands. If the user is combining any of these functions for binning data with his/her own decision about where binning starts (as is done on the blog which is linked to above) the functions above are all incorrect. With an arbitrary starting point for binning 'Min', the correct function is:

```
bin(x) = width*(floor((x-Min)/width)+0.5) + Min
```

You can see why this is correct sequentially (it helps to draw a few bins and a point somewhere in one of them). Subtract Min from your data point to see how far into the binning range it is. Then divide by binwidth so that you're effectively working in units of 'bins'. Then 'floor' the result to go to the left-hand edge of that bin, add 0.5 to go to the middle of the bin, multiply by the width so that you're no longer working in units of bins but in an absolute scale again, then finally add back on the Min offset you subtracted at the start.

Consider this function in action:

```
Min = 0.25 # where binning starts
Max = 2.25 # where binning ends
n = 2 # the number of bins
width = (Max-Min)/n # binwidth; evaluates to 1.0
bin(x) = width*(floor((x-Min)/width)+0.5) + Min
```

e.g. the value 1.1 truly falls in the left bin:

- this function correctly maps it to the centre of the left bin (0.75);
- Born2Smile's answer, `bin(x)=width*floor(x/width)`, incorrectly maps it to 1;
- mas90's answer, `bin(x)=width*floor(x/width) + binwidth/2.0`, incorrectly maps it to 1.5.

Born2Smile's answer is only correct if the bin boundaries occur at $(n+0.5)*\text{binwidth}$ (where n runs over integers). mas90's answer is only correct if the bin boundaries occur at $n*\text{binwidth}$.


answered Oct 25 '13 at 17:37



[ChrisW](#)

409 3 5

+1 for paying attention to detail ... thanks – [Abhinav](#) Oct 26 '13 at 1:42

Do you want to plot a graph like this one?  yes? Then you can have a look at my blog article: <http://gnuplot-surprising.blogspot.com/2011/09/statistic-analysis-and-histogram.html>

Key lines from the code:

```
n=100 #number of intervals
max=3. #max value
min=-3. #min value
width=(max-min)/n #interval width
#function used to map a value to the intervals
hist(x,width)=width*floor(x/width)+width/2.0
set boxwidth width*0.9

#count and plot
plot "data.dat" u (hist($1,width)):(1.0) smooth freq w boxes lc rgb"green" notitle
```

edited Oct 17 '14 at 17:55



[Christoph](#)

20.4k 5 19 31

answered Sep 17 '11 at 11:00



[hszx](#)

602 5 6

-
- 17 instead of figure, it'd have been better to post code. – [user13107](#) Sep 23 '13 at 2:03
-

I have found this discussion extremely useful, but I have experienced some "rounding off" problems.

More precisely, using a binwidth of 0.05, I have noticed that, with the techniques presented here above, data points which read 0.1 and 0.15 fall in the same bin. This (obviously unwanted behaviour) is most likely due to the "floor" function.

Hereafter is my small contribution to try to circumvent this.

```
bin(x,width,n)=x<=n*width? width*(n-1) + 0.5*binwidth:bin(x,width,n+1)
binwidth = 0.05
set boxwidth binwidth
plot "data.dat" u (bin($1,binwidth,1)):(1.0) smooth freq with boxes
```

This recursive method is for $x \geq 0$; one could generalise this with more conditional statements to obtain something even more general.

answered Mar 27 '12 at 15:10

 Alex
61 1 3

We do not need to use recursive method, it may be slow. My solution is using a user-defined function rint instead of intrinsic function int or floor.

```
rint(x)=(x-int(x)>0.9999)?int(x)+1:int(x)
```

This function will give `rint(0.0003/0.0001)=3`, while
`int(0.0003/0.0001)=floor(0.0003/0.0001)=2`.

Why? Please look at [perl int function and padding zeros](#)

answered Nov 17 '12 at 17:24

 JOE
31 1

As usual, Gnuplot is a fantastic tool for plotting sweet looking graphs and it can be made to perform all sorts of calculations. **However**, it is intended to plot data rather than to serve as a calculator and it is often easier to use an external programme (e.g. Octave) to do the more "complicated" calculations, save this data in a file, then use Gnuplot to produce the graph. For the above problem, check out the "hist" function is Octave using
`[freq,bins]=hist(data)`, then plot this in Gnuplot using

```
set style histogram rowstacked gap 0
set style fill solid 0.5 border lt -1
plot "./data.dat" smooth freq with boxes
```

answered Jun 25 '14 at 13:03

 Dai
131 2

I have a little modification to Born2Smile's solution.

I know that doesn't make much sense, but you may want it just in case. If your data is integer and you need a float bin size (maybe for comparison with another set of data, or plot density in finer grid), you will need to add a random number between 0 and 1 inside floor. Otherwise, there will be spikes due to round up error. `floor(x/width+0.5)` will not do because it will create pattern that's not true to original data.

```
binwidth=0.3
bin(x,width)=width*floor(x/width+rand(0))
```

edited Feb 27 '14 at 8:15

 Christoph
20.4k 5 19 31

answered Dec 2 '13 at 14:58

 path4
41 1

1 That makes absolutely no sense! – [Christoph](#) Dec 2 '13 at 16:54

You haven't encountered such situations, but you may later. You can test it with normally distributed integers with a float sd and plot histograms with `bin=1`, and `bin=s.d`. See what you get with and without the `rand(0)` trick. I caught a collaborator's mistake when reviewing his manuscript. His results changed from absolutely nonsense to a beautiful figure as expected. – [path4](#) Feb 27 '14 at 5:07

Ok, maybe the explanation is so short, that one cannot understand it without a more concrete test case. I'll make a short edit of your answer so that I can undo the downvote ;) – [Christoph](#) Feb 27 '14 at 8:14

Consider integers of normal distribution. Since they are integers, many of them will have the same x/width . Let's say that number is 1.3. With `floor(x/width+0.5)`, all of them will be assigned to bin 1. But what 1.3

really means in terms of density is that 70% of them should be in bin 1 and 30% in bin 2. rand(0) keeps the proper density. So, 0.5 creates spikes and rand(0) keeps it true. I bet the figure by hszx will be much smoother using rand(0) instead of 0.5. It's not just rounding up, it's rounding up without perturbation. – path4 Feb 28 '14 at 5:08

With respect to binning functions, I didn't expect the result of the functions offered so far. Namely, if my binwidth is 0.001, these functions were centering the bins on 0.0005 points, whereas I feel it's more intuitive to have the bins centered on 0.001 boundaries.

In other words, I'd like to have

```
Bin 0.001 contain data from 0.0005 to 0.0014
Bin 0.002 contain data from 0.0015 to 0.0024
...
```

The binning function I came up with is

```
my_bin(x,width) = width*(floor(x/width+0.5))
```

Here's a script to compare some of the offered bin functions to this one:

```
rint(x) = (x-int(x)>0.9999)?int(x)+1:int(x)
bin(x,width) = width*rint(x/width) + width/2.0
binc(x,width) = width*(int(x/width)+0.5)
mitar_bin(x,width) = width*floor(x/width) + width/2.0
my_bin(x,width) = width*(floor(x/width+0.5))

binwidth = 0.001

data_list = "-0.1386 -0.1383 -0.1375 -0.0015 -0.0005 0.0005 0.0015 0.1375 0.1383
0.1386"

my_line = sprintf("%7s %7s %7s %7s
%7s","data","bin()","binc()","mitar()","my_bin()")
print my_line
do for [i in data_list] {
    iN = i + 0
    my_line = sprintf("%+.4f %+.4f %+.4f %+.4f
%+.4f",iN,bin(iN,binwidth),binc(iN,binwidth),mitar_bin(iN,binwidth),my_bin(iN,binwid
    print my_line
}
```

and here's the output

data	bin()	binc()	mitar()	my_bin()
-0.1386	-0.1375	-0.1375	-0.1385	-0.1390
-0.1383	-0.1375	-0.1375	-0.1385	-0.1380
-0.1375	-0.1365	-0.1365	-0.1375	-0.1380
-0.0015	-0.0005	-0.0005	-0.0015	-0.0010
-0.0005	+0.0005	+0.0005	-0.0005	+0.0000
+0.0005	+0.0005	+0.0005	+0.0005	+0.0010
+0.0015	+0.0015	+0.0015	+0.0015	+0.0020
+0.1375	+0.1375	+0.1375	+0.1375	+0.1380
+0.1383	+0.1385	+0.1385	+0.1385	+0.1380
+0.1386	+0.1385	+0.1385	+0.1385	+0.1390

answered Jul 24 '14 at 0:43

 Winston Smith

25 4