URL: http://cxc.harvard.edu/ciao3.4/dmbinning.html
Last modified: December 2006

AHELP for CIAO 3.4

dmbinning

Context: dm

Jump to: Description Bugs See Also

Synopsis

The CIAO binning syntax

Syntax

```
[bin nameA=min:max:step]
[bin nameA=min:max:#bins]
[bin nameA=min:max:step,nameB=min:max:#bins]
[bin/f nameA=min:max:step,nameB=min:max:step]
[bin vector_name=step]
[bin ...;weight]
```

Description

This help file is split up into the following sections:

- 1. Introduction
- 2. Binning a table to an image: columns with integer data types
- 3. Columns with real data type
- 4. Binning with range and region filters
- 5. Rebinning images
- 6. Binning with weights
- 7. Matching the binning of an existing file

1. INTRODUCTION

One powerful capability of the CIAO tools is to take a table (for example, an event list) and make an image format file from it on the fly by binning on one or more axes. For example, suppose you have a small table with three columns A, B, C and five rows:

Row	A	B	C
1	1.0	2.1	1.8
2	2.8	1.2	4.2
3	2.5	1.3	4.1
4	3.6	2.0	0.3

5	2.0	2.2	5.1

We can "bin on A and B" by making a two−dimensional histogram, thinking of A and B as spatial coordinates. The DM directive

```
[bin A=0.5:3.5:1,B=0.5:3.5:1]
```

is read "bin A from 0.5 to 3.5 in steps of 1, and B from 0.5 to 3.5 in steps of 1". This generates a 3 x 3 grid in which the lower left bin runs in A from 0.5 <= A < 1.5, and so has center (1.0, 1.0). If we take the table above we obtain the array (as it would be seen in an image display tool)

.	**A= 1**	**A= 2**	**A= 3**
B= 3	0	0	0
B= 2	1	1	0
B= 1	0	0	2

We can make a coarser or finer grid by altering the step size since row 1 falls in bin (1,2), row 2 falls in bin (3,1), and so does row 3 (because the A=2.5 is rounded up), and row 5 falls in (2,2). Row 4 is outside the grid and so is discarded; and because we are binning on A and B, the information in the C column is discarded. This process is how we turn event lists into images, or (in the 1−dimensional case) light curves or spectra.

```
dmcopy "evt.fits[bin x=1:8192:4,y=1:8192:4]" img.fits
dmcopy "evt.fits[bin time=74321940.2:74328100.0:10.0]" lc.fits
dmcopy "evt.fits[bin energy=1000:7000:10]" energy.fits
dmextract "evt.fits[bin pi=1:1024]" spectrum.fits
```

2. Binning a table to an image: columns with integer data types

For columns of integer data type, the directive

```
[bin x=1:1024:2]
```

is taken to be inclusive − that is

```
1 <= x <= 1024
```

− so that the first bin consists of x=1 and x=2.

You can omit any or all of the three values (min,max,step) in the binning specification. The default value of the step size is almost always 1.0 (it is possible to set another default in the data file by adding a TDBINn keyword to the header), so [bin x=1:1024] is the same as [bin x=1:1024:1]. The default values of the min and max are found from the descriptor ranges in the data file (for FITS files, these are the TLMINn/TLMAXn keywords). These default values are displayed when you type 'dmlist filename cols'. If you have a column for which the descriptor ranges are not set, the DM can't guess a default and you must use an explicit range.

If the defaults are present in the file, then [bin x] will select the default values; [bin x=::2] will select the default range with a step size of 2; [bin x=2] is accepted as a synonym for this. To use one default, [bin x=:512] or [bin x=12:] or [bin x=12::1] are all acceptable.

When binning on a vector column such as SKY(X,Y) using the default ranges, you can specify the step size with the shorthand

```
[bin sky=4]
```

which is equivalent to

```
[bin x=4,y=4]
```

Another layer of defaulting is provided by the mission database file cxo.mdb, which currently defines standard binning for PHA spectral extractions. In the current release, the only program that recognizes this file is dmextract (see the defaults parameter of dmextract).

Note that you can use a real−valued binning factor, which may be less than one:

```
[bin x=::0.5]
```

makes a double resolution image.

If you specify a bin size which doesn't divide evenly into the binning range, the last bin will be `incomplete'. For instance,

```
[bin x=1:10:3]
```

creates four bins (1:3),(4:6),(7:9),(10:12), but the final (10:12) bin will only include contributions from rows with x=10.

An alternate syntax to specify the total number of bins instead of the bin size is

```
[bin x=1:1000:#20]
```

which says "bin from 1 to 1000 making a total of 20 equal size bins".

3. Columns with real data types

For columns of real data type, be careful when binning coordinate values. The ACIS sky coordinate space runs from 0.5 to 8192.5, for a total of 8192 pixels; pixel (512,512) runs from 511.5 to 512.5 in each axis. Binning [x=1:3:1] makes only 2 bins: (1.0:2.0) and (2.0:3.0); contrast this with the integer−column case where three bins are made: (0.5:1.5), (1.5:2.5), and (2.5:3.5).

Otherwise, the syntax is the same as for integer columns.

4. Binning with range and region filters

Frequently you may wish to combine filtering (see `ahelp dmfiltering') and binning. For example,

```
dmcopy "evt2.fits[ccd_id=7][bin chip]" chip7.img
```

makes an image in chip coordinates, first filtering the event list to accept only events on ccd 7.

Usually you won't filter on the same columns that you're binning on; it's sufficient to set the range. The following two commands produce the same output file but the first version is preferred:

```
dmcopy "srclist.fits[bin flux=100:200]" flux_hist.fits
dmcopy "srclist.fits[flux=100:200][bin flux]" flux_hist.fits
```

In the CIAO1 release, there was a significant difference: the first of these forms would give you an output image over the full default range of the variable 'flux', instead of just the range of interest 100:200, with zeroes in the image outside that range of interest. Sometimes you may actually want this, when you are planning to sum or match different images. The special syntax bin/full or bin/f reproduces the CIAO1 behaviour:

```
dmcopy "srclist.fits[flux=100:200][bin/f flux]" flux_hist.fits
```

This is particularly relevant when applying sky region filters:

```
dmcopy "evt2.fits[sky=circle(4096,4096,100)][bin sky]" circle1.fits
```

makes a 201 x 201 pixel image with all the pixels outside the circle set to the null value (zero by default, but

see `ahelp dmopt'). However,

```
dmcopy "evt2.fits[sky=circle(4096,4096,100)][bin/f sky]" circle1.fits
```

makes an 8192 x 8192 pixel image (the default range of SKY) with all the pixels outside the circle zero. Also note that binning by different factors in the x and y directions (e.g. [bin x=3,y=4]) while filtering on a region like a circle will give incorrect results.

5. Rebinning images

Once you have binned an event file into an image, it may be necessary to change the original binning. This may be done using the same syntax as shown before:

```
dmcopy "binned.img[bin x=::4,y=::4]" rebinned.img
```

It is also possible to filter and rebin the image at the same time:

```
dmcopy "acis_img2.fits[sky=region(save.reg)][bin sky=1]" \
acis_new_img2.fits
```

The tool dmregrid may also be used to rebin images, and allows for more complicated changes (e.g. rotating the image).

6. Binning with weights

If one appends a semi−colon followed by a column name to any of the above binning specifications, then the binning will be weighted by the specified column. That is, for each row in the table that contributes to the histogram, the appropriate bin will be incremented by the value of the weight column in that row (rather than the default value of 1). For example, suppose one used the syntax

```
[bin x=0.5:2.5,y=0.5:2.5;pha]
```

to bin the following table:

Row	x	y	pha
1	1	2	11
2	1	1	10
3	2	1	200
4	1	2	3

In this case, the output image would look like

.	x= 1	x= 2
y= 2	14	0
y= 1	10	200

Any numeric column (integer or real) can be used as a weight column. By default, the output image has the same data type as the weight column; this can be overridden with an '[opt type=...]' directive (see 'ahelp dmopt'). Array and vector columns cannot be used as weights. However, scalar components of vectors may be used.

7. Matching the binning of an existing file

The "grid()" syntax may be used to match the binning of an existing file. This is frequently done when creating response files, so that the ARF and RMF are created on the same grid:

```
unix% pset mkarf engrid="grid(rmf.fits[cols ENERG_LO,ENERG_HI])"
```

It may also be used in lightcurve creation to have the same TIME bins in different files:

```
unix% pset dmextract \
      "acis_evt2.fits[bin time=grid(lc_1.fits[cols time_min,time_max])"
```

Bugs

See the bugs page for the Data Model library on the CIAO website for an up−to−date listing of known bugs.

See Also

calibration
caldb
chandra
coords, guide, isis, level, pileup, times
chips
chips
concept
autoname, parameter, stack, subspace
dm
dm, dmcols, dmfiltering, dmimages, dmimfiltering, dmintro, dmopt, dmregions, dmsyntax
gui
gui
modules
paramio, pixlib, stackio
slang
overview, slang, tips
tools
dmcopy, dmextract, dmfilth, dmgroup, dmimghist, dmregrid, fullgarf, get_sky_limits, mkacisrmf, mkarf, mkexpmap, mkgarf, mkgrmf, mkinstmap, mkpsf, mkrmf, mkwarf, tgextract

---

URL: http://cxc.harvard.edu/ciao3.4/dmbinning.html
Last modified: December 2006