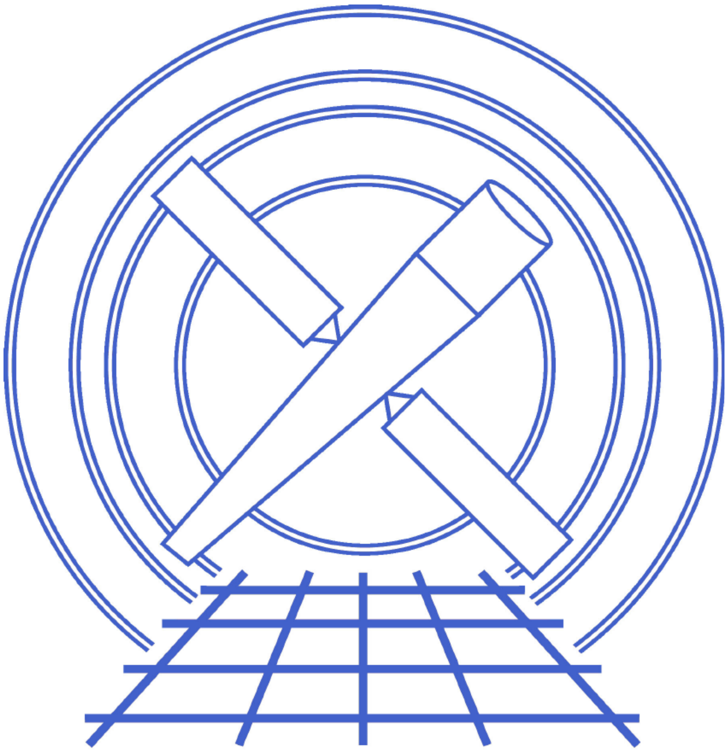
CXC-DM-011

CXC Data Model



Chandra Data Model Manual, Part 2: Toolkit User Guide

CIAO 2.2 Edition
Chandra X-ray Center
October 22, 2001

Contents

1 Introduction

This document describes how to use the basic CIAO data filtering and binning toolkit, consisting of the programs dmlist, dmcopy and dmextract. It has significant overlap with the on-line ahelp files for those tools. Users may wish to first read the User Introduction which gives simple examples, and to use this document in conjuction with the User Reference which describes the full DM syntax.

2 Using dmlist to inspect files

2.1 Syntax and description

```
dmlist infile opt [outfile] [rows] [cells]
```

‘dmlist’ dumps the contents or header of a file or block (a block is a subfile or FITS extension) to ASCII in an organized way. It corresponds to the FTOOLS fdump and fstruct programs, but interprets the input file at a higher level. All CXC data model formats are supported.

dmlist uses a comma-delimited list of options to select which information is displayed: blocks, keys, comments; Header, Cols, Subspace, Data, Array, Full, Struct, all, clean, raw.

Selected rows of data may be dumped: rows=min:max

2.2 Examples

(1) dmlist acis.fits full,all — more

List everything dmlist can tell you about the file and display the output using ”more”.

(2) dmlist acis.fits opt=blocks

See what blocks are in the file ”acis.fits”. (Since opt is a positional parameter, you can omit the ’opt=’ if you like).

(3) dmlist ”acis.fits[2]” opt=cols

See what columns are in the second block (note that we always count starting at 1).

(4) dmlist ”acis.fits[events]” opt=header,subspace

Look at the header and the data subspace for the block called ”events” in the file. You can choose blocks either by name or by number. If you don’t specify a block, the program guesses which one you want.

(5) dmlist ”acis.fits[events][pha=20:30]” opt=data rows=100:104

Look at rows 100 to 104 of the virtual file filtered to show only pulse heights between 20 and 30. Note the row numbers are those passing the filter, not the original row numbers of the underlying file.

(6) dmlist ”pha2.fits[cols channel,counts]” opt=array rows=3:3

Look at the channel and counts arrays for row 3 of the file, displayed in ‘array’ (vertical) format.

(7) dmlist ”acis.fits[events]” opt=header,raw verbose=0

Look at the FITS header instead of the DM-level header (the ’raw’ option specifies use of the low level interpretation) and generate a minimal, ‘bare’ version of the output (verbose = 0).

Some more detailed examples:

```
 dmlist file.dat blocks
```

In our test file, this gives the result

```
--------------------------------------------------------------------------------
Dataset: file.dat
--------------------------------------------------------------------------------

     Block Name                          Type         Dimensions
--------------------------------------------------------------------------------
Block    1: HDU0                           Null
Block    2: EVENTS                         Table          20 cols x 6933     rows
Block    3: GTI                            Table           2 cols x 3        rows
```

This tells us the file has 3 blocks, consisting of a null block followed by 2 tables, the first of which has 20 columns and 6933 rows.

In general, the CXC tools operate on a single ‘block’ (table or image) in the file. If you want to look at the header for a block called EVENTS quote the block name in square brackets following the filename.(Actually, in this case, EVENTS is the default block in the file so you could get away with leaving it out).

```
 dmlist "file.dat[events]" header
```

This won’t list all of the FITS header keys, just the ones that aren’t ‘structural’. The data model recognizes keywords like NAXIS (the number of axes in an image) or TUNIT4 (the unit string for the fourth column in a table) as part of the structure of the file, and doesn't count them as header metadata. In contrast, keywords like INSTRUME (the name of the instrument) or AVG_ROLL (some generic piece of information written by the creating program) are separate data in themselves, and while they are associated with the data in the table or image, they don't describe its structure. The dmlist header command lists only these extra, non-structural header keys. The information contained in the structural keys is presented by other options to dmlist: the cols option, which describes columns in a table, and the subspace option which describes how the file has been filtered.

To see the data without the header,

```
dmlist "file.dat[events]" data
```

This will produce several thousand lines of output, so you probably want to either redirect the output to a file or specify an explicit output file:

```
dmlist "file.dat[events]" data outfile=tmp1
```

Alternatively, you can ask to see only the first 10 rows:

```
dmlist "file.dat[events]" data rows=1:10
```

```
--------------------------------------------------------------------------------
Block    2: EVENTS                            Table          20 cols x 6933     rows
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
Data for Table Block EVENTS
--------------------------------------------------------------------------------

ROW    TIME           MJF        STARTMNF STOPMNF CRSV       CRSU

     1 95469560.428006       2617       5       5          80          13
     2 95469560.554886       2617       5       5          42           3
     3 95469560.681366       2617       6       6         169          13
     4 95469560.920710       2617       7       7         102           8
     5 95469561.320374       2617       9       9          77           2
     6 95469561.656374       2617      11      11         151           6
     7 95469562.140630       2617      12      12         142          10
     8 95469562.289910       2617      15      15          99           8
     9 95469563.145110       2617      16      16          99           8
    10 95469563.339750       2617      16      16          99           8
```

(the remaining columns are suppressed here for clarity).

To list the columns in the event file and the units, type, and range of the data;

```
dmlist file.dat cols
```

The largest amount of information is provided by

```
dmlist "file.dat[events]" full
```

which is shorthand for

```
dmlist "file.dat[events]" blocks,header,cols,subspace,data
```

You can combine any dmlist options by separating them with commas.

If you omit the block name, e.g.

```
 dmlist "file.dat" data
```

the program will give you the first block in the file (except that it will ignore a null primary header in a FITS file). You can also ask for a block by number, remembering that in the CXC software things are always numbered starting at 1, so that

```
 dmlist "file.dat[2]" data
```

gives the data for the second block in the file.

Because dmlist uses the data model interface, you can also use the full filtering capability described later in this manual, for instance to see only the TIME and PHA columns for time values in a particular 2-second period:

```
dmlist "file.dat[events][time=95470577.0:95470579.0][cols time,pha]" data
--------------------------------------------------------------------------------
Block    2: EVENTS                         Table           2 cols x 12       rows
--------------------------------------------------------------------------------



--------------------------------------------------------------------------------
Data for Table Block EVENTS
--------------------------------------------------------------------------------

ROW    TIME           PHA

     1 95470577.055270        124
     2 95470577.083366         99
     3 95470577.145014        124
     4 95470577.156406        226
     5 95470577.188102        107
     6 95470577.457638        134
     7 95470577.525878        178
     8 95470577.528742        145
     9 95470577.894214         92
    10 95470578.012118        135
    11 95470578.094582         97
    12 95470578.431686         87
```

Note that the program now reports the EVENTS block as having only 2 columns and 12 rows; it sees only the filtered view of the table, not the full table itself.

2.3 Parameters

2.3.1 infile

The input virtual file specification. (string, input, required)

2.3.2 opt

The list of options, separated by commas. **Note:** the options are a single parameter to the tool. When listing multiple options directly on the command line, there must be no spaces between them. Thus,

```
dmlist file.dat header,cols
```

is good, and so is

```
dmlist file.dat opt="header, cols"
```

but *not*

```
dmlist file.dat header, cols
```

since this is read as two separate tool parameters by the parameter interface library.

Acceptable options are:

- **blocks** - Summarize all the blocks (images or tables) in the file, one line per block. The dimensions of the image, and the number of rows and columns in the table, are given for each block; the name of the block is also given, and you can use this name to access specific information about the block.
- **keys** - List the ASCDM header keys for the selected block. Remember that not all the FITS keywords in a FITS file will be included - the ones like EXTNAME and CRPIX

that have a special meaning for the structure of the file don't count as ASCDM keys, their values show up in other places (e.g. the output from cols). With this option, the comment and history records are suppressed. To see the low level list of all raw FITS header keywords in a FITS file, use opt=header,raw.

- **comments** - List the ASCDM comment keys for the block. For a FITS file, these keys are the COMMENT and HISTORY keywords.
- **header** - Equivalent to Keys,Comments. List both DM level header keys and comment/history records. - List the ASCDM header keys for the selected block. Remember that not all the FITS keywords in a FITS file will be included - the ones like EXTNAME and CRPIX that have a special meaning for the structure of the file don't count as ASCDM keys, their values show up in other places (e.g. the output from cols). To see the low level list of raw FITS header keywords in a FITS file, use opt=header,raw.
- **cols** - List the ASCDM columns for the selected block. Shows 'vector columns' like (X,Y) as pairs. This output shows you the names of the variables you can filter a table on and what their valid ranges are. If the block is an image, it gives you one column with the name of the image (images are interpreted as a table with a single row and column).
- **subspace** - Summarizes the data subspace for the block. This describes the filters that have been applied to the data, either by the user or in processing.
- **data** - Prints the data segment. The cells parameter controls how much of an array column gets printed. By default, the data is printed in an 'ornate' format showing arrays grouped in parentheses etc. If you use "Clean" option as well, the output is in a simpler format suitable for reading by other analysis programs. Known bug: for images, the attempt to print a whole array on a single line can cause the software to crash - use 'array' instead.
- **array** - Prints the data segment, but with arrays printed vertically. The cells parameter controls how much of an array column gets printed.
- **full** - Equivalent to 'blocks,header,cols,subspace,data'
- **struct** - Equivalent to 'header,cols,subspace'
- **all** - Print the info selected by the other options for all the blocks in the dataset.
- **clean** - When used in conjunction with the Array or Data options, produces stripped down output suitable for parsing by other programs.
- **raw** - Provides a lower level view of the header. 'header,raw' lists the raw FITS header keywords, not just the data model keywords. The old 'data,raw' functionality is now provided by 'data,clean'.

2.3.3 outfile

The output ASCII file to be created; output is sent to the screen if this parameter is blank.

2.3.4 rows

Range of table rows to print (min:max)

```
rows=30:40
```

Print row 30 to row 40

```
rows=40
```

Print row 1 to 40

```
rows=40:40
```

Print only row 40

```
rows=" "
rows=-1
```

Print all rows

2.3.5 cells

Range of array indices to print in array columns and in images.

```
cells = 1:8
```

Print only array elements 1 to 8 of each array

```
cells = all
```

Print all array elements.

2.3.6 verbose

Standard CIAO parameter. Controls amount of information to print (0-5). The verbose parameter provides debugging information; verbose = 0 is usually fine.

2.3.7 mode

Standard CIAO parameter. See the parameter interface documentation. mode=h will stop the program prompting for parameters.

3 dmcopy

```
dmcopy infile outfile [kernel] [option] [verbose] [clobber] [mode]
```

'dmcopy' is a very general program which copies a 'virtual file' (a table or image with a filter specification) to a physical disk file. For details of the virtual file syntax, see "ahelp dmsyntax". Some uses are:

- Filtering tables. dmcopy selects rows and columns from a table, making an output file consisting of the filtered table.
- Binning tables to an image. Select any n columns from a table, and make an n-dimensional binned image from them.
- Filtering images, selecting subsets (spatial, temporal, energy, etc.) of an image.

When do you get a table as output and when do you get an image? If the input to dmcopy is an image, the output will be an image. If the input is a table and you use the "bin" or "opt image" commands, you'll get an image as output. Otherwise, you'll get a table.

3.1 Examples

(1)

```
dmcopy "bas.fits[stdevt]" out.fits
```

This copies the table STDEVT in dataset "bas.fits" to "out.fits". Note that use of the Unix command line requires quotes when the string has [ ] in it. In the DM paradigm, any GTI (Good Time Interval) table in bas.fits is also copied, as it contains information that's considered to be "part of" the event table.

(2)

```
dmcopy "bas.fits[stdevt][time=5000:5200]" out.fits
```

This copies the table STDEVT in dataset "bas.fits" to "out.fits", including only those rows with times between 5000 and 5200. This assumes that STDEVT contains a column called TIME. We also copy along the Good Time Intervals table, if present, and update it by intersecting with our time filter. If present, ONTIME, LIVETIME and EXPOSURE keywords are also updated.

(3)

```
dmcopy "bas.fits[stdevt][time=5000:5200]" out.fits option=all
```

The use of the "option=all" command in dmcopy forces any other tables or images in the bas.fits dataset to be copied over along with the table you are filtering. For instance, a ROSAT FITS event file may have a STDQLM table, among others. The default behaviour of dmcopy is to drop such extra tables during a copy, but option=all ensures they are carried along.

(4)

```
dmcopy "bas.fits[stdevt][pha=20:30,grade> 4]" tmp1.fits
dmcopy "bas.fits[stdevt][time=1522012:1522320,1522400:1522600]" tmp2.fits
dmcopy "bas.fits[stdevt][(x,y)=circle(3001.2,4982.3,14.5)]" tmp3.fits
dmcopy "bas.fits[stdevt][sky=circle(3001.2,4982.3,14.5)]" tmp3.fits
dmcopy "acis.fits[events][status=10x1x10]" tmp3.fits
dmcopy "region.fits[2][shape=point]" tmp3.fits
```

These examples filter the same table selecting only some events. (You can use the dmlist tool to find out what columns are available for filtering). Support has been added in CIAO1.1.5 for filtering on bit type columns (using a bit string of ones and zeros with 'x' used as a wild card) and on string type columns - see the last two examples above.

(5)

```
dmcopy "input.fits[EVENTS][bin x=10:100:1,y=1:100:1]" image1.fits
```

This bins an event list in input.fits to create an image in ”image1.fits”. The qualifier [bin x=10:100:1, y=1:100:1] selects a subset of the image with no blocking. To ”block” the image, use e.g. [bin x=10:100:2,y=1:100:2], giving an image each of whose pixels corresponds to 4 pixels in the original image.

(6)

```
dmcopy "input.fits[events][bin pha=1:4096]" phaimage.fits
```

This bins an event list in input.fits to create a 1D PHA image: (see the dmextract tool for binning it to an XSPEC compatible PHA table file)

(7)

```
dmcopy "image.fits[200:512,128:500]" image3.fits
```

This copies only that part of the image for which the first axis pixel number lies between 200 and 512 and the second axis pixel number is between 128 and 500.

3.2 Parameters

3.2.1 infile

Input virtual file. The CXC virtual file syntax is filename[qualifier][qualifier]....; see ‘ahelp dmsyntax’ or the syntax guide in this manual for details.

Currently FITS and IRAF/QPOE,IMH format files are supported. If the virtual file is not well defined, the program gives an error message and exits. Qualifiers that can be used to filter the input file are:

[block]: the block part of the syntax is the table/image name (for FITS, HDU name or 1-based HDU number). You can omit the [block], in which case the first ‘interesting’ block is used: in the case of FITS, null primary headers and good time intervals are ignored.

filter - is a list specifying ranges of column values to accept, filtering out rows of a table or portions of an image. In the case of an image, where the axes may be unnamed, the ’name=’ can be omitted and the ranges are assumed to be in order of the axes.

cols - specifies the names of columns to include, when selecting from a table.

bin - is a sequence defining table columns to bin to create an image file.

newblock - is a name for the resulting block; if omitted, the name of the input block is used.

3.2.2 outfile

The output filename to be created.

3.2.3 kernel

The output file format type (fits/iraf). Defaults to the same as input.

3.2.4 option

option=value (image/all/type). Controls the behaviour of dmcopy.

- option = all, Copies all blocks in the input file, not just the one you are filtering. The other blocks are copied without change. However, GTIs are treated specially.
- option = image, Forces the virtual file to be interpreted as an image, even if the input is a table and there is no [bin] element in the specification.
- (deprecated) option = ”type=i4”, used with binning tables to images. Forces the type of the resulting image to be 4-byte integer. ”type=i2” (2-byte integer, which is the default anyway) and ”type=r4” (4-byte real) are also supported. This option is now supported within the virtual file syntax itself, e.g. ”dmcopy ”file.fits[bin sky=4][option type=i4]”, and is provided here for back compatibility. Note that this option does not work when copying images to images, and so cannot be used to change the type of an existing image.

3.2.5 verbose

Verbose can be from 0 to 5, generating different amounts of output.

3.2.6 clobber

If outfile already exists, clobber=yes will allow you to overwrite it. The default is no.

3.2.7 mode

Standard CXC parameter; mode=h suppresses parameter prompting. See the parameter interface documentation.

4 Using dmcopy to filter tables

The **dmcopy** tool is a versatile program which you can use to manipulate data. Unlike **dmlist**, which produces text output, **dmcopy** produces a new data file in one of the supported formats. It provides a way to filter files by copying a virtual file to disk. In simple use, you type:

```
dmcopy virtual-file output-file
```

In this section we describe the syntax you can use to define a virtual table which is a filtered version of the input table. See the reference manual chapter for full details of the syntax. The simplest case of a virtual file is the name of a complete file:

```
dataset_name
```

This gets you the first 'interesting' block in the dataset. In a FITS file, this is deemed to be the first block for which NAXIS (the number of image dimensions) is nonzero. That means that you get the image in a FITS image file, and the first binary table in a FITS binary table file with a null primary HDU.

For analogy with FTOOLS, we provide the following syntax to access the nth block in the dataset:

```
dataset_name[n]
```

where n is a positive integer. Example:

```
rh1012.fits[2]
```

accesses the 2nd block in the file rh1012.fits. (note that throughout the DM we always count starting at 1; the FITSIO subroutine library counts HDUs from 1, but the FTOOLS programs count from 0).

A friendlier syntax for accessing a specific block is:

> dataset_name[block_name]

In a FITS file, the block name is defined to be the value of the HDUNAME keyword, if present. otherwise it's the value of the EXTNAME keyword (concatenated with EXTVER if that is present). If none of these are present, the blocks are named HDU1, HDU2, etc. You can use

> dmlist dataset_name blocks

to find the names of the blocks in a file.

4.1 Filtering on ranges

We select rows by

> dataset_name[block_name][name=min:max,min:max,min:max,
> name=min:max,min:max...]

for example:

```
dmcopy "rh1012.fits[events][pha=1:20,50:90,time=100:200,500:900]" outfile.fits
```

The columns may also be accessed by column number e.g.

```
dmcopy "rh1012.fits[events][#1=1:20,50:90,#4=100:200,500:900]" outfile.fits
```

4.2 Filtering on spatial regions

Regions are explained more fully in section **??**. As an example, consider a simple region, a box centered at detector pixels 4100,3170 and with sides of length 25 pixels.

```
dmcopy "infile.fits[events][(tdetx,tdety)=box(4100,3170,25,25)]" outfile.fits
```

To extract events from the above box changed to width 200, height 30, and rotated 45 degreees (as seen projected on the sky):

```
dmcopy "infile.fits[events][(tdetx,tdety)=rotbox(4100,3170,200,30,45)]"
outfile.fits
```

For a source within a circular region of radius 7 pixels and centered at sky coordinates 4732,4385;

```
dmcopy "infile.fits[events][(x,y)=circle(4732,4385,7)]"  outfile.fits
```

For convenience the notation, sky = (x,y), may be used in Chandra event files; use ’dmcopy filename cols’ to see what names the DM recognizes for pairs of columns in a particular file.

Regions may be combined. For a 45-90 degree sector of an annulus;

```
dmcopy "infile.fits[events][sky=annulus(4732,4385,10,100)*pie(4732,4385,100,45,90)]" out
```

You can import text region files and CXC FITS region files with the syntax

```
dmcopy "infile.fits[events][sky=region(filename)]" outfile.fits
```

Regions may be excluded by (in interactive mode) preceding the shape name with a ”!”; on the unix command line this must also be preceded by a backslash. Thus to exclude the above source;

```
dmcopy "infile.fits[events][exclude (x,y)=\!circle(4732,4385,7)]"  outfile.fits
```

however, this does not work for the ”region(filename)” option. A more general approach is to use the filter exclude syntax:

```
dmcopy "infile.fits[events][exclude (x,y)=circle(4732,4385,7)]"  outfile.fits
dmcopy "infile.fits[events][exclude (x,y)=region(ds9.reg)]"  outfile.fits
```

Rotated ellipses and other regions are also supported. See section **??**.

In addition, it is possible to define grids of regions, using the “pgrid” and “rgrid” specifiers, which are based on the pie and rectangle region specifiers respectively. Each ‘cell’ in the grid is then effectively an input file for the DM tool. The syntaxes are

pgrid(xcen,ycen,rmin:rmax:dr,thetamin:thetamax:dtheta), and

rgrid(xmin:xmax:dx,ymin:ymax:dy).

The second variable, theta or y, is the more rapidly varying variable. That is, the order in which the cells are stepped through is determined by taking the starting value of the first variable (r or x) and stepping through all the values of the second variable (theta or y), then incrementing the first variable. For example,

foo.fits[events][sky=rgrid(1:3:1,2:6:2)]

expands to an input stack

foo.fits[events][sky=rectangle(1.0,2.0,2.0,4.0)]
foo.fits[events][sky=rectangle(1.0,4.0,2.0,6.0)]
foo.fits[events][sky=rectangle(2.0,2.0,3.0,4.0)]
foo.fits[events][sky=rectangle(2.0,4.0,3.0,6.0)]

(the syntax for rectangle is (xmin,ymin,xmax,ymax))

and

foo.fits[events][(x,y)=pgrid(10,10,0:2:1,0:360:180)]

expands to an input stack

foo.fits[events][(x,y)=pie(10.0,10.0,0.0,1.0,0.0,180.0)]
foo.fits[events][(x,y)=pie(10.0,10.0,0.0,1.0,180.0,360.0)]
foo.fits[events][(x,y)=pie(10.0,10.0,1.0,2.0,0.0,180.0)]
foo.fits[events][(x,y)=pie(10.0,10.0,1.0,2.0,180.0,360.0)]

(pie syntax is (xcen,ycen,rmin,rmax,thetamin,thetamax)).

To use the grid input above, one would need to specify four output files in an output stack. The output stack file is simply an ascii file with the names of the output files one to a line. For example:

```
gridout1.fits
gridout2.fits
gridout3.fits
gridout4.fits
```

If the output stack is called "outfiles.lis", then, putting it all together, the dmcopy command would be:

```
dmcopy "foo.fits[events][sky=rgrid(1:3:1,2:6:2)]" @outfiles.lis
```

which is equivalent to the four commands

```
dmcopy "foo.fits[events][sky=rectangle(1.0,2.0,2.0,4.0)]" gridout1.fits
dmcopy "foo.fits[events][sky=rectangle(1.0,4.0,2.0,6.0)]" gridout2.fits
dmcopy "foo.fits[events][sky=rectangle(2.0,2.0,3.0,4.0)]" gridout3.fits
dmcopy "foo.fits[events][sky=rectangle(2.0,4.0,3.0,6.0)]" gridout4.fits
```

4.3 Column selection and renaming

We can select columns from a table block by

> dataset_name[columns column-list]

for example

```
rh1012.fits[columns det,pha,time]
```

In the select qualifier, there are two special reserved column names: #all and #none, with synonyms * and -. Thus

```
rh1012.fits[events][columns -]
```

makes a copy of the header and data subspace of rh1012.fits[events], but with an empty table/image section. This is useful for making a filter file (see below). (Note: the data subspace records the selection history of the data, e.g. GTI, PHA filter, etc: see the data model design document).

To delete a column, use the ! operator:

```
rh1012.fits[columns !pha]
```

Finally we can specify the desired name of the virtual file, although this may be overriden by some applications:

```
rh1012.fits[events][columns pha,time][newname]
```

The required ordering of the qualifiers is intended to reflect a logical ordering of the operations: first an input block is chosen, then it is filtered on rows, then on columns, and finally it is associated with a named virtual block. Explicitly naming the virtual block may be useful in cases where the program is expecting a block with a particular name, e.g.

```
rh1012.fits[EVENTS][STDEVT]
```

opens the table EVENTS, but when the program asks what the table's name is, the data model will 'lie' and tell it that the name is 'STDEVT'.

5 Using dmcopy to bin tables into images

The primary dataset used by X-ray astronomers is the photon event list, which is a table containing information about one photon in each row. To visualize this event list, users will want to bin it into an n-dimensional image. The virtual file syntax gives you a way to pick any set of columns in the event list, or any other table, and bin them up into a histogram. The simplest example is to make an image from the X and Y sky pixel position columns. The table may look like

```
POS(X,Y)        Range = 1:5
X       Y       TIME     PHA  Other columns....
4       3       10       3    ...
4       4       15       3    ....
4       3       15       3    ...
1       2       20       2    ....
```

meaning that in the above 5 x 5 pixel space, we have detected 4 photons: 2 of them at (4,3), 1 at (4,4) and 1 at (1,2). Choosing a (default) binning factor of 1, we can make a 5x5 X,Y image as follows:

```
    X 1  2  3  4  5
  Y----------------
  1 | 0  0  0  0  0
  2 | 1  0  0  0  0
  3 | 0  0  0  2  0
  4 | 0  0  0  1  0
  5 | 0  0  0  0  0
```

Note that the event list representation is more compact for this sparse image, but we want to be able to make the image representation so that we can look at it in standard image display programs, and use it in existing image analysis software.

Alternatively, we may not care about the spatial dependence of the photons, but instead want to look at the light curve. The process is the same but only in 1 dimension: suppose we bin by 5 time units and represent the results by a new table:

```
TIME  |
----  |
 0-5  | 0
 5-10 | 1
10-15 | 2
15-20 | 1
```

we now have a one-dimensional image of counts versus time. We could do the same with PHA (pulse height, related to photon energy),

```
PHA   |
----  |
1     |  0
2     |  1
3     |  3
4     |  0
```

The **dmcopy** program allows users to carry out these binning operations and write the output to an image file, usually a FITS image. As for the TIME and PHA examples above, it can also be useful to store a 1-D image as a new, binned, table; the information is the same, but the representation is different. The standard analysis programs for PHA spectra require the binned data to be stored as a table rather than an image, so that extra columns like statistical error can be stored at the same time; the **dmextract** program described elsewhere in this manual is used to bin into a table format file. The **dmcopy** program is only able to make an image format file, so you'll need to use **dmextract** to make an XSPEC-compatible PHA file.

5.1 Explicit binning of tables

To make an X,Y image from an event list – the most common thing you might want to do – simply use dmcopy as follows:

```
 dmcopy "rh1012.fits[bin x=32,y=32]" image.fits
```

This makes an image whose pixels are binned by 32 in each direction (if rh1012.fits has a range of x and y of 1:8192, this will compress to a 256X256 image). To make a full resolution image of a small section,

```
dmcopy "rh1012.fits[bin x=4000:4200,y=4000:4200]" image2.fits
```

The "bin x,y" syntax denotes that we are binning instead of filtering, and specifies the columns we want to bin on.

If we want to make an image on columns which are not the default columns, we use the syntax

```
rh1012.fits[bin pha,time]
```

This is the replacement for the old PROS key= syntax. The fact that the keyword 'bin' is followed by a space and not = allows us to recognize that this grouping is a binning command and not a filter on a column named 'bin'. We may also want to support the 'key=' syntax as a back compatibility option.

These binning operations on tables use a bin size of 1 and use the TLMIN/TLMAX keywords to figure out the size of the resulting image. If you want to use a different bin size and axis range, the syntax is

```
dmcopy "rh1012.fits[events][bin pha=4:10:2,time=100:200:10]" outfile_img.fits
```

which makes a 2D image of counts versus pha and time, with the pha axis having 4 pixels running from 4 to 10 in steps of 2, and the time axis having 10 pixels running from 100 to 200 in steps of 10. This syntax syould also be used if the keywords are not there (as is the case with some old archival files).

```
rh1012.fits[bin energy=5:10:0.02]
```

will make a 1-D image of counts versus energy (i.e. a spectrum). In general

```
block_name[bin name=min:max:step,name=min:max:step,...]
```

If min is greater than max, the output image has the axis reversed with respect to the input one.

To filter and bin at the same time,

```
rh1012.fits[events][pha=2:7,y=1000:1100][bin pha=4:10:2,time=100:200:10]
```

Note that this makes a 2-D pha, time image which is 4 x 10 pixels in size, but the pixels with pha more than 7 will be zero since those pha values were filtered from the table.

To block a table into an image by a given factor,

```
rh1012.fits[bin detx=4,dety=4]
```

which is equivalent to the old

```
rh1012.fits[key=detx,dety][bl=4]
```

The bl=n blocking option would now by default be interpreted as a filter on a column called 'bl'. Again, back compatibility support is a possibility.

For a 'vector column' (seen in dmlist as DET(DETX,DETY)) you can use just the grouped column name, "det", or "sky" for (x,y), or "chip" for (chipx,chipy).

```
rh1012.fits[bin det=4]
```

5.2 Implicit binning of tables

The dmcopy program tries to interpret the syntax as a table first, and only if that fails does it try to open an image. You can force it to try and make an image with the 'option=image' parameter. Other programs may explicitly require an image as imput. If the program is trying to open an image, but you give it a table (possibly with a filter applied), what does it do? As a special dispensation, we guess that you really meant to bin the image in the default way. The 'default way' has several levels to it:

- The most basic default is to bin the table on its first two columns with a bin size of 1.
- If the table has defined 'preferred axes', those columns are used instead of the first two. These preferred axes are defined in the CPREF keyword of a FITS file.
- If there are no explicit preferred axes, but the table has columns called X and Y, those columns are used.

Example:

```
dmcopy "rh1012.fits[events]" image.fits option=image
```

makes an image by binning rh1012.fits in the default way.

A row-filtered table:

```
rh1012.fits[events][pha=4:8,time=100:200,400:500]
```

may also be used as input to a virtual image; in this case the rows are filtered prior to image binning. If the base block is an image, the same syntax implies an image section on the named axes.

6 Using dmcopy to filter images

The **dmcopy** program and other DM programs can also be used to filter images – in other words, to make a new image by selecting a subset of the pixels in that image.

The same simple syntax we used to refer to tables:

```
ih1012.fits
ih1012.fits[1]
ih1012.fits[events]
```

may also refer to an image.

6.1 Image filtering

The syntax for image filtering is the same as for table filtering. Currently you can only image filter on 2 axes, and in many images the axes don't have names. Images made from event lists by dmcopy do retain the original column names as axis names.

6.2 Image filtering: IRAF notation

We also support the old IRAF image section notation, i.e.

```
ih1012.fits[100:200,100:400]
```

which is equivalent to

```
ih1012.fits[#1=100:200,#2=100:400]
```

7 Using dmcopy to format output

In this section we describe the syntax you can use to define the format of the output file. At this time two formats are supported: fits and iraf. The format is set by the "kernel" parameter.

7.1 FITS to IRAF

To extract a small region from a fits events table and output an iraf qp file;

```
dmcopy "infile.fits[events][x=3500:4500,y=3025,4025]" outfile.qp kernel=iraf
```

To extract an iraf image at full resolution:

```
dmcopy "infile.fits[events][bin x=3500:4500:1,y=3025:4025:1]" outfile.img kernel=iraf
```

Unfortunately, the PROS software makes some ROSAT-specific assumptions which mean that Chandra QPOE data can only be used by certain PROS tasks; timing tasks have been used with some success.

8 Using dmcopy to rebin images

To rebin an image, we use the same syntax as for binning tables. However, support for rebinning images is still unreliable if the images are blocked (image pixel coarser than original pixel).

```
rh1012.fits[bin pha=4:10:2,time=100:200:10]
```

if this time rh1012.fits is a 2D pha versus time image, a new lower resolution image is generated with the same axes but coarser pixels.

9 dmextract

9.1 Description

Make a histogram table file (e.g. PHA file) from a table column. Generate count histogram on supplied regions for a spatial table or image file.

```
dmextract infile outfile [bkg] [error] [bkgerror] [bkgnorm] [exp]
[bkgexp] [sys_err] [opt] [defaults] [wmap] [clobber] [verbose]
```

'dmextract' creates a histogram from a column of data in a table. Both scalar (PHA, time, etc.) and vector (DET, SKY, etc.) columns are supported. dmextract thus includes the capability to create PHA files, light curves, and spatial radial profiles.

In the case of scalar columns, a simple linear binning of the data is available, while for vector columns ('spatial extraction') the user can define the bins via a set of regions. Support for background correction is available for both scalar and vector columns. Vector column extraction also supports exposure corrections and images as input.

9.2 Extraction on a scalar column

'dmextract' takes as input a table or set of tables, often an event list. It generates an 'extraction', a histogram of the data binned on any one column in the table. The resulting file is also a table, containing the histogram and associated errors and rates. For a scalar column the extraction is defined using the CXC Data Model (DM) binning syntax (see "ahelp dmsyntax") in the input file string (see examples).

The default mode of dmextract ("opt=pha1") is to generate a HEASARC/OGIP-compatible Type I PHA file. For X-ray event data, the PHA (pulse height) is an instrumental quantity related to the photon energy; the mapping from PHA to energy varies with detector position and sometimes time, and we also use the PI (pulse invariant) value, which is the PHA corrected to a standard energy scale. The PHA files are used in combination with response matrices (RMF files), which are made for either PHA or PI binning. See the threads on spectral fitting for advice on how to use PHA files.

Using "opt=generic" is appropriate when binning on a column other than PHA or PI. This generates a file which is similar to a Type I PHA file, but doesn't contain keywords specific to PHA data. For instance, you can bin on TIME to make a simple light curve, or you can bin an aspect solution file on RA to see the histogram of the RA distribution.

Using "opt=pha2" makes a Type II PHA file, in which each line of the file contains a complete spectrum with errors. This option is used when the input is a stack of virtual files. One line of the output file is created for each input extraction, so the infile in this case will usually be a stack. For columns other than pha and pi, opt=generic2 makes a similar file without PHA specific keywords.

The complete binning specification is "[bin col=min:max:step]". For instance,

dmextract "evt1.fits[bin pha=1:2048:2]" out.pha

You can leave out any of the min,max,step, for instance

[bin pha=:2048], [bin pha=::2], [bin pha=5::]

Default values will be filled in from the maximum valid range of the column (in the case of a FITS file, the values of the TLMINn, TLMAXn columns) and the default binning (the

CXC specific keyword TDBINn, if present). If all values are omitted, e..g [bin pha], and the optional "defaults=filename" parameter is supplied, dmextract will look in the defaults file for a suitable binning for that column.

Both types of output files will have the columns CHANNEL, COUNTS, STAT_ERR, and COUNT_RATE. Type II output files will also have the columns SPEC_NUM and TOTCTS.

It can be useful to encode extra information about the extraction in the output file. For the ASCA mission, the idea of a WMAP (weight map) was introduced: associated with the extraction table is an image of the extraction region. Specifically, the PHA FITS file has in its primary header an image containing a low resolution map of the source in detector coordinates. This allows downstream software to determine the appropriate weighting for calibrations which depend on detector position (for instance, effective areas may depend on the off-axis angle). The dmextract wmap parameter allows the user to define such a wmap; we recommend 'wmap="det=8"' to create a map in DETX,DETY coordinates binned by a factor 8, if you are making a PHA file which will be used with mkwarf (see "ahelp mkwarf"). The wmap option is currently only useful when making a PHA file, although in principle it could be used by software making exposure calculations for other quantities.

'dmextract' optionally writes a SYS_ERR keyword to the file header, using the value of the sys_err parameter provided by the user. The SYS_ERR keyword denotes the fractional systematic error associated with the data; thus "sys_err=0.05" indicates a 5 percent error. This error may be added in quadrature by downstream software to the statistical errors provided in the extraction.

Error conditions:

- If either the requested column does not exist, the minimum bin number exceeds the maximum bin number, or the step size is negative, an appropriate error message is displayed and the program halts.
- If the input file does not contain a LIVETIME keyword, livetime is set to 1.0.

9.3 Defaults File

The dmextract defaults file is compatible with the XSELECT mission database (mdb) file. It is a text file containing lines of the form telescop:instrume:key value where telescop and instrume are the values of the FITS TELESCOP and INSTRUME keywords, and dmextract recognizes keys of the form colname_binning. Example:

Chandra:ACIS:pi_binning 0:4095:2

For example, to add a default for HRC time binning of 20s bins, one might add

Chandra:HRC:time_binning ::20.0

9.4 Extraction on vector column - spatial binning

The second method of extraction for 'dmextract', binning on a 2-dimensional quantity like 'sky(x,y)' or 'det(detx,dety)' can take an event or image file and a set of regions to extract and sum the counts in each of those regions. The user can supply a background file or regions to remove from the source to create a total count rate. There are several options for the extraction syntax.

```
filename[bin sky=shape(x0,y0,...)]
```

Indicates a count extraction on the sky column with a single bin specified by the shape (for example circle(4096,4096,20)). The user may supply a stack of regions, with each region a separate line in an ASCII file region.lis:

```
filename[bin sky=@region.lis]
```

or, for the special case of the ANNULUS shape, can define a set of nested annuli with

```
filename[bin sky=annulus(x0,y0,rmin:rmax:step)]
```

(this syntax will only work in a command line, not in a region file).

Note that in no cases are default binning parameters allowed for vector column extraction (e.g., filename[bin sky] or filename[bin det] are not allowed and will result in a fatal error). A valid region specifier must be included (e.g. sky=circle...).

The source extraction will count up the number of events within each region. Errors will be calculated on the counts based on either poisson or gaussian statistics. The count rate is determined by using the exposure time for the file. If a user supplies a background file with regions, or a fixed background (counts/pixel/s), this value will be used to perform a background correction on the source counts. The information on the background counts will be included in the output as well as net counts and net rates. Background information can be normalized. The default is to normalize the counts with respect to the relative exposure times and geometric areas in the input and background files.

The user can optionally supply an exposure map that corresponds to the input file. The regions chosen in the input will be extracted and a weighted exposure correction over that region will be applied to the counts. The background file can also have an exposure correction applied to it. An EXPOSURE (and BG_EXPOSURE if relevant) column will be added to the output table; note that the EXPOSURE keyword will no longer be present in the header.

The are three different methods of error calculations. Errors are generally calculated by poisson methods using the Gehrels approximation:

$$\text{error} = 1 + \sqrt{(}N + 0.75)$$

Alternatively, the user may select gaussian errors:

$$error = \sqrt{(}N)$$

If the user has a variance image at the same pixel scale as the input file, this can be used instead of other error methods. The region from the input file is extracted from the variance image, and the sum of the variances in the region is used for error calculations.

The region areas are expressed in units of physical pixels and accounts for the binning in the case of an input image file.

9.5 Examples

(1)

```
 dmextract infile="input.fits[sky=circle(4095.0,4096.0,43.0)][bin
pha=1:3000:10]" outfile=output.pha clobber=yes verbose=2
```

Perform a scalar extraction on the single input file input.fits, taking photons from a circle around a particular source position specified in sky pixel coordinates, and binning on column PHA from channels 1 to 3000 with a step size of 10, and output to a type I PHA output file output.fits.

(2)

```
dmextract  infile="input.fits[sky=annulus(4095.0,4096.0,60.0,100.0)][bin
pi=::100]" outfile=output.pha clobber=yes verbose=1
```

Perform a scalar extraction on the single input file input.fits, binning on column PI from channels TLMIN to TLMAX (both read from input.fits) with a step size of 100, and output to a type I PHA output file output.fits. Include only counts within the specified sky coordinate annulus.

(3)

```
dmextract infile="input.fits[sky=annulus(4095.0,4096.0,60.0,100.0)][bin pi]"
outfile=output.pha clobber=yes verbose=1 defaults=cxo.mdb
```

Same as above, but using the mission default from file cxo.mdb:

```
cat > inlist
in1.fits[bin detx=1000:1500:10]
in2.fits[bin detx=1000:1500:10]
^D
cat > outlist
out1.fits
out2.fits
^D
dmextract infile=@inlist outfile=@outlist clobber=yes opt=generic
```

Perform extractions on the input files in1.fits and in2.fits listed within the ASCII file inlist, binning on column DETX from channels 1000 to 1500 with a step size of 10, and output to corresponding type I output files out1.fits and out2.fits listed within the file outlist:

(4)

```
dmextract infile=@inlist outfile=out.pha2 clobber=yes opt=generic2
```

Identical to the previous example except output to the single type II file out.fits.

(5)

```
dmextract infile="input.fits[bin sky=circle(4095.0,4096.0,43.0)]"
outfile=output.fits bkg=none error=poisson clobber=yes verbose=2
```

Performs a single source count (vector) extraction in a circle centered at 4095,4096 with a radius of 43. There is no background correction specified and the errors reported are poisson.

Note that because sky is a vector (2D) column, using the syntax ”bin sky” means this is a count extraction. Since only one region is specified, the output table will have a single row, the counts within that region.

(6)

```
dmextract infile="input.fits[bin sky=annulus(4095.0,4096.0,0.0:100.0:10)]"
outfile=output.fits bkg=none error=poisson clobber=yes verbose=2
```

Performs a spatial source count (vector) extraction in a set of annuli centered at 4095,4096 with radii 0, 10, 20, 30,.. 100 as specified by the 0.0:100.0:10 binning declaration. There is no background correction specified and the errors reported are poisson. The result is a radial intensity profile table.

(7)

```
dmextract infile="input.fits[bin sky=circle(4095.0,4096.0,43.0)]" outfile=output.fits
bkg="input.fits[bin sky=circle(4000.0,4096.0,40)]"
error=gaussian bkgerror=gaussian bkgnorm="1.0" clobber=yes verbose=2
```

Perform a single source count extraction on the input file input.fits, counting photons from a circle around a particular source position specified in sky pixel coordinates, and background subtracting the normalized counts from the background region. Gaussian errors are reported for the counts and rates.

(8)

```
dmextract infile="input.fits[bin sky=@region.lis]"
outfile=output.fits bkg=none exp="expmap.fits" clobber=yes verbose=2
```

Where region.lis is a file that contains 1 region per line, e.g.:

```
circle(4233.5,3753.5,28)
circle(4233.5,3753.5,18)
circle(4233.5,3753.5,10)
circle(4233.5,3753.5,5)
```

Performs a spatial source count extraction on the input file into bins specified by the region list. The exposure map is opened and a weighted exposure correction is found for each region and applied in the calculation of the net counts.

(9)

```
dmextract infile="input.fits[bin sky=@region.lis]"
outfile=output.fits error="variance.fits" clobber=yes verbose=2
```

Performs a spatial source count extraction on the input file. Errors for the counting are determined by the variance image "variance.fits". For each region, the variance in the total counts extracted is the sum of the variances in the corresponding pixels in the variance map. The number of output bins is equal to the number of regions in region.lis.

9.6 Parameters

9.6.1 infile

The input virtual file or stack, e.g. event list, modified by a dmextract binning command.

Any table or stack of tables is valid input, with the

```
table[bin scalar_col=min:max:step]
```

type of extraction or the

```
table[bin vector_col=region_list]
```

type of extraction.

Images can also be input with the latter type of extraction:

```
image[bin vector_axis=region_list]
```

There are three methods of stack inputs allowed.

- PHA/Histogram extraction TYPE I files:

  For each file in the input, there should be a file specified in the outfile stack.

- PHA/Histogram extraction TYPE II files:

  There can be multiple input files, but only one output file. This will create a type II file as described above.

- Radial Profile/Source Extraction:

  For each file in the input, there should be a file specified in the outfile stack. However, if only 1 output file is specified, dmextract will place all output information in the single file with the header information for the first input file.

9.6.2 outfile

The output histogram file.

The output file. This is a histogram of counts and rates for each bin (a bin is a PHA channel in the case of a pha file, histogram bin in the case of a generic scalar extraction, or a region in the case of a vector, spatial, extraction).

Please see the infile parameter for a detailed description of outfile stacks.

9.6.3 bkg

Background file with regions or a numeric value in units count/pixel/s. For spatial extraction only.

This is a string that is one of:

- a background virtual file (defining a single background region that will be used for each extraction bin)
- a background file stack (defining a set of regions to be used for each extraction bin; the number of entries in the stack must be the same as for the input stack in this case)
- a numeric value that is in units count/pixel/s

9.6.4 error

The error method for determining error on the counted items. Default: gaussian.

These are the methods for error determination. The current options are "poisson","gaussian", or a variance image. A variance image is not yet supported in histogram or pha binning, instead gaussian will be used.

9.6.5 bkgerror

The error method for the background error on the background count.

These are the methods for background error determination. The current options are "poisson","gaussian", or a variance image. A variance image is not yet supported in histogram or pha binning, instead gaussian will be used.

9.6.6 bkgnorm

Background normalization for spatial count extraction.

This fudge factor allows the user to adjust the background normalization relative to the source normalization in the calculation of net counts, for a spatial extraction. When the factor is 1.0, the background counts are normalized by the ratio of exposure times and spatial areas. If bkgnorm is different from 1, the normalized background counts are further multiplied by the value of bkgnorm.

9.6.7 exp

Exposure map for the input file for spatial extraction. Can be a stack.

The exposure map that corresponds to the input file will have the information in the input regions extracted and the weighted exposure correction over this region is applied to the counts. If a stack of exposure maps is supplied, there must be the same number of files as the infile stack. The tool will report an error otherwise. The units of the exposure map are either cm**2 sec, cm**2 (normalized by time), or dimensionless. Exposure map and background exposure map units must be compatible. Otherwise, an error will be reported and the calculation will continue without the exposure correction. For Chandra exposure maps generated with the standard CIAO tools MKINSTMAP and MKEXPMAP, the source and background exposure maps must BOTH be either unnormalized (units of cm**2-sec) or normalized (units of cm**2). In either case, the net counts reported by DMEXTRACT will be in units of counts/cm**2.

9.6.8 bkgexp

Exposure map for the background file for spatial extraction.

The exposure map that corresponds to the background file will have the information in the input regions extracted and the weighted exposure correction over this region is applied to the counts. If a stack of background exposure maps is supplied, there must be the same number of files as the bkg stack. The tool will error otherwise. The units of the exposure map are either cm**2 sec, cm**2 (normalized by time), or dimensionless. Exposure map and background exposure map units must be compatible. Otherwise, an error will be reported and the calculation will continue without the exposure correction. For Chandra exposure maps generated with the standard CIAO tools MKINSTMAP and MKEXPMAP, the source and background exposure maps must BOTH be either unnormalized (units of cm**2-sec) or normalized (units of cm**2). In either case, the net counts reported by DMEXTRACT will be in units of counts/cm**2.

9.6.9 opt

The output file type format (PHA1/PHA2/GENERIC/GENERIC2). Default is PHA1.

There are two main kinds of output generated by dmextract. The 'generic' output is a histogram table with each row corresponding to a different bin. The 'generic2' output is a set of histograms, one per row, with the histograms stored in array columns.

Most users will want 'opt=generic' for most applications. However, when doing a spectral extraction on PHA or PI, you need to use 'opt=pha1' or 'opt=pha2' to generate a compliant PHA type I or II file for use with Sherpa or XSPEC.

9.6.10 defaults

The mission database file; default is $ASCDS_CALIB/cxo.mdb

The mission database file gives default binning instructions for certain extractions; for instance, it knows that PI extractions on ACIS files should use the binning 1:1024:1, instead of the default binning 0:1024:1 that dmextract would assume from the TLMIN/TLMAX values in the file header if no mission database file were specified.

The MDB file is just a text file, and you can generate your own MDB file to define your preferred defaults for various extractions.

9.6.11 wmap

The binning to use to make a WMAP.

If you specify a wmap binning with "infile=inval wmap=wmapval", dmextract will make an image in the output file header which will be equivalent to the image generated by 'dmcopy "inval[bin wmapval]" wmapfile'. If you specify "wmap=default", a value from the mission database file specified by the "defaults" parameter will be used. For Chandra ($ASCDS_CALDB/cxo.mdb) this is equivalent to 'wmap="chip=32"'. We suggest that 'wmap="det=8"' be used if you wish to use the file as the input to mkwarf.

9.6.12 clobber

Specifies if an existing output file should be overwritten.

9.6.13 verbose

Specifies the level of verbosity (0-5) in displaying diagnostic messages.

10 Using dmextract to create PHA files

The **dmextract** program is intended to be similar to using **dmcopy** to bin tables into images, but instead of creating an image format file it will write the binned data to a tabular histogram. **dmextract** may be used to make histograms of all kinds, but has a special mode which supports making HEASARC-style, XSPEC-compatible PHA files.

The dmextract tool generates an 'extraction' which is a histogram of the data in the input file together with histograms of the counting errors and the counting rates, using the binning information. Binning is currently limited to a single column.

In a 'type I' file, the histogram's bin ranges and the counts per bin are written 'vertically', as scalar columns in the output table. In a 'type II' file, multiple histograms are created and written 'horizontally', with the ranges and counts per bin as arrays in a single row, and each row representing a different histogram.

If either the requested column does not exist, the minimum bin number is greater than the maximum bin number, or a data value lies outside the range of the minimum and maximum bins, an appropriate error message is displayed and the program halts. If a value for the livetime is not in the header, a warning message is displayed, and the livetime is given a default value of 1.0. Currently, the default output consists of an OGIP Type I PHA file with columns channel, counts, statistical error, and counting rate. Type II PHA files and generic histogram files can also be made.

10.1 Examples

The following command extracts the spectrum from a specific region after filtering on event grade and pha. See sections 4 and **??** for the various filtering options.

```
dmextract infile="marx.fits[stdevt]
[sky=circle(4096,4024,15),grade=0,2:4,6,pha>30][bin PHA=1:3000:10]"
 outfile=output.evt clobber=yes
```

To prepare Chandra data for use with Xspec:

```
dmextract "infile.fits[events][bin PHA=1:4096:2]" outfile.fits
dmextract "infile.fits[events][bin PHA]" outfile.fits defaults=cxo.mdb
```

To make a histogram of counts versus CCD column number on a set of rows on chip 7,

```
dmextract "evt.fits[events][ccd_id=7,chipy=256:512][bin chipx=1:1024]"
   histo.fits opt=generic
```

10.2 Using *dmextract* to generate spectra from multiple sources

Using the feature of 'stack' input and output, it is possible to extract spectra from multiple sources. In this section we assume all the sources are from the same events list, but it is equally possible to specify a different file for each source — the tool does not care, since the spectra are not extracted simultaneously. The 'stack' format runs *dmextract* as many times as there are lines in the input stack.

Assuming that there are n sources, there are two ways of isolating them. One could display the entire events list using an imager, e.g. SAOtng or ds9, and then graphically select each source and save the corresponding region specification as an ascii file. The other option is to use DM's own region specifications described in section **??**.

Using the first method, one would obtain n region files, called, say, source1.reg, source2.reg, ... , source$n$.reg. The input stack file, which we will call "instack", will then look like (assuming binning on the PI column)

event.fits[sky=region(source1.reg)][bin pi]
event.fits[sky=region(source2.reg)][bin pi]
event.fits[sky=region(source3.reg)][bin pi]
...
...
event.fits[sky=region(source$n$.reg)][bin pi]

with one source specification per line.

Using the second, the input stack file would contain something like

event.fits[sky=circle(4000,4240,10)][bin pi]
event.fits[sky=circle(3780,3988,6)][bin pi]
...
...
event.fits[sky=circle(x_n,y_n,rad$_n$)][bin pi]

For the output, there are two options: either to generate separate files each containing the spectrum from one source, i.e. type I PHA files, or to generate one type II PHA file containing all the spectra. In the former case it will be necessary to create an output stack file, which we will call "outstack",containing the names of the output files, one to a line, corresponding one-to-one to the input file specifications. *dmextract* can be run on each input, output pair by

```
unix:  dmextract @instack @outstack opt=pha1
```

If creating a type II PHA file an output stack is not required, only an output file name:

```
unix:  dmextract @instack outfile=spectra.pha opt=pha2
```

As a final note, it should be remembered that the only requirement is that each line in the input stack is a valid virtual file specification. It is not necessary that they be parts of the same file, or have the same filtering.

10.3 Parameters

The parameters for dmextract are described below,

- **infile:** The first is the input block, which should be a table (e.g. an event list). If the file exists, and is readable, then the user is queried for the next parameter.
- **outfile:** The second parameter is the output dataset name.
- **opt:** can have values 'pha1, 'pha2', 'generic', 'generic2'. Defines style of output.
- **defaults:** Specifies a mission database file, e.g. $ASCDS_CALIB/cxo.mdb. This defines a default pha binning for the instruments.
- **clobber:** Specifies if an existing output file should be overwritten.
- **verbose:** Specifies the level of verbosity in displaying diagnostic messages.
- **mode:** Standard IRAF mode parameter.