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

AHELP for CIAO 3.4

dmfiltering

Context: dm

Jump to: Description Bugs See Also

Synopsis

The CIAO filtering syntax

Syntax

```
[filter nameA=min1:max1,min2:max2,...minN:maxN]
[filter nameA=min1:max1,...minN:maxN,nameB=min1:max1,...]
[nameA=min1:max1,...minN:maxN,nameB=min1:max1,...]
[nameA=val,nameB=val1,min2:max2,val3,,...minN:maxN]
[filter nameA=shape(parameters),nameB=REGION(region_filename)]
[filter nameA<max,nameA>min,nameA!=val]
[filter (nameA=min1:max1)||(nameA=min3:max3,nameB=min3:max3)]
[exclude nameA=min1:max1,min2:max2,..,nameB=...]
[filter @filter.lis]
```

Description

This help file is split up into the following sections:

- 1. Table filtering on columns with real data types
- 2. Table filtering on columns with integer data types
- 3. Table filtering on columns with character string data types
- 4. Table filtering on columns with bit data type
- 5. Table filtering on columns with logical data type
- 6. Table filtering on vector columns using region filters
- 7. Compound filters
- 8. Exclude filters
- 9. Filters defined in files: using the @ syntax

1. Table filtering on columns with real data type

There are a number of ways to filter a DM block (table or image in a file). In this section we describe table filtering; see also `ahelp dmimfiltering' for filtering files which are in image format. To see which column names you can filter on, use

```
unix% dmlist a.fits cols
```

to display the columns in the main block of file a.fits.

The simplest kind of filter is a range filter

```
[filter energy=1000:2000]
```

which specifies that the DM should only see rows in the input file which satisfy 1000.0 <= energy < 2000.0. You can use this filter by appending it to a filename or block name in any of the CIAO tools:

```
unix% dmcopy "a.fits[filter energy=1000:2000]" b.fits
unix% dmcopy "a.fits[events][filter energy=1000:2000]" b.fits
```

Note that the prefix "filter" is optional:

```
unix% dmcopy "a.fits[energy=1000:2000]" b.fits
```

You can filter on multiple quantities:

```
unix% dmcopy "a.fits[energy=1000:2000,time=5410300:5410320]" b.fits
```

You can also filter on multiple ranges for each quantity:

```
unix% dmcopy "a.fits[energy=1000:2000,4000:8000,grade=0,2:4,6]" b.fits
```

This filter accepts rows which have energies between either 1000 and 2000 or 4000 and 8000, and grades equal to 0, 2 to 4, or 6. Note that you can leave out the colon if the min and max of a range are the same, so 0:0 becomes just 0. If you want to express "less than" and "greater than", you can just retain the colon but omit the min or max, so

```
[energy=:4000]
```

means accept energies up to 4000; whilst

```
[detx=:511,513:]
```

means accept all values of detx except 511.0 <= detx < 513.0.

2. Table filtering on columns with integer data types

The interpretation is a little different depending on whether the table column has integer or real data type. For an integer data type column,

```
[energy=4000:5000]
```

means (4000 <= energy <= 5000), in other words both ends of the range are included.

The syntax also supports the special pseudo−column #row, the row number of the unfiltered file:

```
[#row=100:200]
```

3. Table filtering on columns with character string data types

Filtering is a little more restricted with character string columns. You can only use the colon syntax, for example:

```
[filter shape=m:n,point,rectangle,s:z]
```

Be careful: the range m:n includes everything beginning with m, and it includes the letter n, but not other strings beginning with n: for example, "ngc" is not within m:n since in an ASCII ordering ngc > n. The comparison is case−sensitive.

4. Table filtering on columns with bit data type

Columns with bit data type are a special case. The most common example is the STATUS column in the event files, which is 32 bits wide for Chandra ACIS event files generated between launch and at least 2003 (it may be increased at a later date). Suppose for simplicity you instead have a status column with only 6 bits, and wish to accept rows with the 3rd bit from the end set and the end bit equal to zero. A simple numeric filter of the type described above would have to be very complicated to describe all the numeric values for which these bits have the desired values; instead, the user supplies a `bitmask string':

```
[filter status=xxx1x0]
```

The string may only contain the characters 1 (corresponding bit must be set), 0 (bit must not be set) and x (wild card: bit may have any value).

The bit pattern display convention here is that the rightmost bit displayed (with the value 0 in the example above) is the least significant bit, which is bit 32 in the usual FITS convention and bit 0 in the usual C programmers convention.

5. Table filtering on columns with logical data type

Logical data columns contain a value of either true (1) or false (0). When filtering on a "true" value, any of the following will work:

```
unix% dmlist "srclist.fits[double=1]" data
unix% dmlist "srclist.fits[double=T]" data
unix% dmlist "srclist.fits[double=TRUE]" data
unix% dmlist "srclist.fits[double=true]" data
```

Likewise, for "false" values:

```
unix% dmlist "srclist.fits[double=0]" data
unix% dmlist "srclist.fits[double=F]" data
unix% dmlist "srclist.fits[double=FALSE]" data
unix% dmlist "srclist.fits[double=false]" data
```

6. Table filtering on vector columns using region filters

In CIAO, a `vector column' is a paired column consisting of two components. For instance, the DET vector column has components DETX and DETY. You can see which of the columns in the file are vector columns by using `dmlist filename cols'. There are two ways to filter on a vector column: define a rectangular region by filtering on each of the components as usual,

```
unix% dmcopy "evt.fits[detx=4000:5000,dety=3000:4000]" rectangle.fits
```

or, use a region filter (see `ahelp dmregions' for the full region syntax) on the vector column, either using its name − in this case DET − or the two components in parentheses − in this case (DETX,DETY).

```
unix% dmcopy "evt.fits[det=circle(4500,3500,120)]" circle.fits
unix% dmcopy "evt.fits[(detx,dety)=circle(4500,3500,120)]" circle.fits
```

7. Compound Filters

The syntax also supports compound (logical OR) filters:

```
unix% dmcopy \
 "evt.fits[(ccd_id=2,chipx=512:513)||(ccd_id=7,chipx=500:520)]" \
 two_bits.fits
```

Note that we do not support arbitrarily complex C−style logical expressions, just lists of filters separated by ||.

8. Exclude filters

A very useful feature is the ability to invert a filter, so excluding rows or pixels instead of including them:

```
unix% dmcopy "evt.fits[exclude sky=region(reg.ds9)]" holes.fits
unix% dmcopy "evt.fits[exclude pha=2:100,grade=7]" clean.fits
```

This is particularly useful in conjunction with compound filters (see the previous section):

```
unix% dmcopy \
 "evt.fits[exclude (ccd_id=0:6,8:9,chipx=513)||(ccd_id=7,chipx=512:513)]" \
 better.fits
```

9. Filters defined in files: using the @ syntax

Filter specifications may be applied from a file rather than including them on the command line. This is especially useful when applying the same filters to multiple input files. The filter filename is preceded by an "at" symbol (@).

```
unix% dmlist "acis_evt2.fits[@filters.lis]" counts
```

where filters.lis contains:

```
sky=rotbox(4148.125,4043.625,7.58978,22.338761,44.516094),
pi > 100
```

Note that each complete filter must be separated with a comma, just as they would be on the command line. This syntax can also be used to apply the GTIs (Good Time Intervals) from one file to another, so:

```
unix% dmcopy "evt2.fits[@gti.fits]" evt2_filtered.fits
```

filters evt2.fits using the GTIs stored in gti.fits.

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, dmbinning, dmcols, dmimages, dmimfiltering, dmintro, dmopt, dmregions, dmsyntax

gui
: gui

modules
: paramio, pixlib, stackio

slang
: overview, slang, tips

tools
: acisspec, dmappend, dmarfadd, dmcopy, dmextract, dmgroup, dmgti, dmjoin, dmlist, dmmerge, dmpaste, dmsort, dmtcalc, dmtype2split