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

AHELP for CIAO 3.4

dmgti

Context: tools

Jump to: Description Examples Parameters ALGORITHM CHANGES IN CIAO 3.4 Bugs See Also

Synopsis

Create a Good Time Interval from limits placed on time−based files (event and Mission Time Line files).

Syntax

```
dmgti infile outfile userlimit [mtlfile] [lkupfile] [smooth] [clobber]
[verbose] [mode]
```

Description

The dmgti tool creates a Good Time Interval (GTI) filter file which represents the times of rows in the input file which pass the user−supplied constraints. As an example, the tool can be used to find out those times when the RATE column of a lightcurve lies below a set limit; the resulting output file can then be used to filter out from an event file those times where the rate was above this limit.

The input file (the infile parameter) can be any table file that can be read by the Data Model and contains a column called time which is stored in ascending order. Event files or Mission Time Line (MTL) files are examples of valid input. For accurate results the time column should be in constant size time bins with the appropriate TIMEPIXR and TIMEDEL keywords.

How to write constraints

The syntax for the contraints − given as the userlimit parameter − is the same as used for dmtcalc. See "ahelp syntax" for details of this syntax. Extra constraints can also be supplied using the mtlfile parameter; this feature is mostly used in the Chandra pipelines and will not be needed by most users. The format of the file is described in the help for the lkupfile parameter ("ahelp −b PARAM −t lkupfile dmgti").

Unlike dmtcalc and dmimgcalc, dmgti is primarily interested in boolean statements, since it needs to know whetehr a particular row passes or fails the supplied constraints. Therefore, the userlimit parameter will be set to statements like

```
((rate>2.0)&&(rate<5.0))
```

− which means only allow times for which the RATE column lies between 2.0 and 5.0.

Output files

There are two outputs. The main output is the set of times which pass the constraints, written out as a CIAO−style GTI filter file which you can use to filter other data files. This file contains an empty datamodel table called 'FILTER' with a GTI table attached to it (in FITS, as a separate HDU). The GTI table is also compatible with other analysis systems such as FTOOLS. If called gti.fits it can be viewed with:

```
  unix% dmlist gti.fits subspace
```

If the output file already exists, it is combined with the new GTI information. For instance, if the output file is an existing event file with a GTI attached, it is modified in place to intersect the new GTI with the old GTIs. The event data themselves will be unaltered. Howeverm we recommend that you do not do this and instead create a new file and use dmcopy directly for combining it with other files.

The optional second output is a copy of the timeline file containing only those columns used in the constraints − including any new ones created from the combination of existing columns − with an extra column added called "limit_status". The data type of this column is a 1−bit bitmask, and is 1 for rows which FAIL the constraints and 0 for rows which PASS the constraints (the good times). If the userlimit contained any smoothed columns (e.g. "rate:5") then the smoothed values will be written out to this file unless the smooth parameter is set to "no".

NOTES

- If the task fails for any reason, it returns a −1.
- Byte columns are read as shorts, which are converted to doubles due to the smoothing application.
- New columns that are defined should have the datatype of the expression they are set equivalent to, but this has not been as thoroughly tested as it ought. Use with caution.
- Brackets − ie ()'s − should be used in complicated expressions to ensure values are calculated in the correct order.

Example 1

```
dmgti lc.fits gti.fits userlimit="rate<3"
```

Creates a GTI file which lists as "good" times only those times for which the rate column of lc.fits − here assumed to be a lightcurve − is less than 3. Note that, unless the expression is surrounded by ()'s − as is done in the next example, spaces should not appear in the userlimit expression.

The resulting GTI file can then be used to filter an event file − e.g. evt2.fits − by saying

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

The Filtering Lightcurves thread contains more examples of this.

Example 2

```
dmgti mtl.fits gti.fits userlimit="( (volt1:3>9.2) && (volt1<13.0) )"
```

Here the GTI file contains only those times for which the vol1 column of mtl.fits − after smoothing by 3 rows − is

greater than 9.2 and less than 13.0. Since the volt1 column is smoothed in one of the expressions, the smoothed value is used in all expressions.

The whole expression is surrounded by () to ensure that the spaces do not cause problems (see the "WHITESPACE IN EXPRESSIONS" section of "ahelp syntax" for more information).

Example 3

```
dmgti mtl.fits gti.fits userlimit="((volt1:3>9.2)&&(volt1<13.0))"
mtlfile=smooth_mtl.fits lkupfile=mtl_limit.fits smooth=yes
```

The same run as above, but also using a pipeline−style FITS limits file and recording the smoothed output in an output mtl file.

Parameters

name	type	ftype	def	reqd
infile	file	input		yes
outfile	file	output		yes
mtlfile	file	output	none	
userlimit	string	input		yes
lkupfile	file	input	none	
smooth	boolean		yes	
clobber	boolean		no	
verbose	integer		0	

Detailed Parameter Descriptions

Parameter=infile (file required filetype=input)

This is the input timeline.

The timeline (MTL) table should contain a time ordered 'time' column and a TIMEDEL keyword giving the size of the time bins in the same unit as the values in the column, as well as columns corresponding to all the variables you will supply constraints for.

Parameter=outfile (file required filetype=output)

Output GTI file

This is the file that the GTI will be written to. If the filename provided exists, then the GTI will be written to that file, updating the data subspace that exists within it (the GTI will be applied as a time filter, and the new GTIs will be the intersections of it with the old GTIs). If the file is a new one, it will be created.

Parameter=mtlfile (file filetype=output default=none)

Output smoothed/filtered MTL file

This file is a copy of the input MTL file that may have had individual smoothing factors applied to the column data. The file will also be a subset of the input MTL, unless ALL the columns of the input MTL were filtered on in creating the GTI. In the case where all columns determine the GTI and no smoothing is applied, it will be a copy of the input. The purpose is to provide the user with the actual data that the GTI was created against. If this is set to NONE, none or "", the file won't be created.

Parameter=userlimit (string required filetype=input)

Optional user defined limit string

This is the constraint expression defining the GTI. Each record in the timeline file is checked against this expression to see if its time will be included in the output GTI.

Parameter=lkupfile (file filetype=input default=none)

Lookup table defining which MTL columns to check against

This is the lookup table defining which MTL columns to check against. Most users will leave this parameter as "none" and just use the `userlimit' parameter to set the constraints.

Format of lookup table

If used, the lookup file must be a datamodel (e.g. FITS) table with a single text column called "gti_limits", up to 256 bytes in size. Each line of this table should be a constraint in the same format as used by the userlimit parameter. This file can also contain a special keyword, GTICLASS, of type string. The value of keyword HDUCLAS2 in the output GTI is set to this. If the keyword does not exist, then a default value of "STANDARD" is used.

The algorithm section at the end of this document describes how the limits are used.

Parameter=smooth (boolean default=yes)

Smooth the input MTL data?

If smooth is set to no, smoothing factors in the constraints expression are ignored.

Parameter=clobber (boolean default=no)

Clobber output file if it exists?

Parameter=verbose (integer default=0)

Debug level.

Debug levels range from 0 (nothing) to 5, which will describe in detail the processing that is occuring.

ALGORITHM

The program checks that each variable mentioned in the constraints corresponds to a column in the timeline file (or to a new column alias defined elsewhere in the string). If the column is not found, or if a numeric column is given a string−valued constraint, an error is issued.

If the constraints are well−posed, the tool loops over each row of the timeline, smoothing the values if required (see below), and evaluating the row against the constraints.

The limits conditions define the number of rows of data over which a given column will be smoothed. These rows are centered about the row which is to be smoothed for, i.e., for the third row of a column with a smoothing factor of 5, rows 1−5 will be smoothed. In the case of an even smoothing factor, it is promoted to the next odd integer. Edge effects are handled by truncation − for instance, for a column with smoothing factor 3, row 1 is the average of rows 1 and 2 instead of 0, 1 and 2, since there is no row 0.

CHANGES IN CIAO 3.4

Using TIMEPIXR keyword

dmgti now properly uses the TIMEPIXR keyword to modify the TIME column when creating GTI files from lightcurves. Users may see a small small shift in the time filter when compared to CIAO 3.3 because of this change.

Parameter File

The kernel parameter has been removed.

Bugs

See the bugs page for this tool and for the mathematical syntax support on the CIAO website for an up−to−date listing of known bugs.

See Also

chandra
mtl
dm
dmcols, dmfiltering, dmopt
tools
dmappend, dmarfadd, dmgroup, dmimgcalc, dmjoin, dmmerge, dmpaste, dmsort, dmtcalc, dmtype2split, mtl_build_gti, syntax

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