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

AHELP for CIAO 3.4

mtl_build_gti

Context: tools

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

Synopsis

Create Good Time Interval from Mission Time Line and limits

Syntax

```
mtl_build_gti infile outfile mtlfile [userlimit] [lkupfile] [smooth]
[clobber] [verbose]
```

Description

The mtl_build_gti tool creates a Good Time Interval from an input Mission Time Line file. The elements of the MTL that the GTI is created from is determined by a limits file, in a FITS file format, and an optional user provided limits string. An optional output MTL is created, which is the subset of columns of the input MTL that the GTI was determined from. As there is an option to perform smoothing on the data, the output MTL will be smoothed, if that option is selected. The input MTL will remain unchanged. An output GTI file will either be created, or if it is an already existent EVENT file, the GTI will be written to it. The GTI filtering syntax is explained in detail in the algorithms section. This format should be followed for both the contents of lkupfile, and the value of the userlimit parameter.

DATA FORMATS

- MTL (Input and Output):
- The data will be contained in the first extension of the FITS file, which will be a BINTABLE. The EXTNAME keyword value of this extension will be MTL. All the columns will be of type double, and besides the potential standard header keywords, there will be a TIMEDEL keyword, whose value is the length of the MTL records, in seconds. The only guaranteed column is time, which will be the first one.
- GTI Output: the output GTI file may be written to a file that already exists. Otherwise, a new output file is created.
- Limits Lookup FITS File: This is a FITS file with one column, gti_limits, this is an ASCII column, of length up to 256 characters. This is to allow for lengthy limits conditions, although it is recommended that the user specify conditions for individual column on separate lines. The name of the extension containing the limits is LIMITS. Integers, real numbers, bools, bit arrays, bytes and string columns can all be checked against. Real numbers may use e/E scientific notation. There are some rules for various datatypes though. String values (not column names) should be surrounded by quotes. For boolean values, 0, 1, T or F are allowed. The T or F do not need to be quoted as strings. For bit arrays the syntax is as follows:
- column_name==XNT|F

- X denotes a bit, N is the bit number (starting from 0), and then you specify T or F depending on whether you want it on or not. You can specify a list of comma delimited XNT|F's. As an example:
- col==X0F,X2F,X5T
- This would specify that the first and second bits should be false, and the 6th bit should be true. It doesn't matter what the other bits are. You can specify that all the bits should be false by X−1F, and that they should all be true by X−1!F. Finally, if you specify "col==0", then the code will check that all bits are not set.
- The following constructs are also allowed within the limits file:
- () && || == != < > <= >= ! * + − / fabs(), sqrt(), ln(), log(), exp(), cos(), sin(), tan(), acos(), asin(), atan(), cosh(), sinh(), tanh()
- In addition, there are two other, important, constructs.
- Column Values:
- To indicate a column value should be inserted, denote the column as follows:
- string:<integer value> _or_ string: _or_ string
- The integer value is the smoothing factor that will be applied. If it is not specified, one will be assumed. While the colon was once necessary to denote a column rather than a string value, that is no longer the case. The colon is only necessary if you wish to specify a smoothing factor other than one. The "string:" notation is a legacy from earlier versions of the tool. You can not perform limit checks against the same column with different smooth factors, i.e., if you reference the same column twice, but with different smoothing factors, the first smoothing factor will be used. As an example, say that you want to allow a column to fall within a certain range or have a certain discrete value outside that range. If the column was counts, and you wanted it to be smoothed by a factor of 3, you would specify it as follows:
- (((0.0 <= counts:3) && (counts: <18.0)) || (counts == 23.0))
- The parenthesis, while not always necessary, are provided for clarity. As another example, lets define a limits check for counts relative to cosine of another column, declination (this doesn't make much sense...):
- ((counts:3 != cos(declination))
- This indicates that the value of counts, smoothed over three rows, should not be equal to the cosine of the unsmoothed value of declination.
- New Columns:
- To indicate a new column, denote it as follows:
- string=expression
- A new column will be created, whose value is the value of the expression. This could simply be another column, or some relationship of columns, constants, and functions. There is no constraint on what datatypes may be copied (previously, the new column was forced to type double, thus constraining the columns it could be derived from). The new column can be referenced, after the initial "=" definition, using the normal column notation − a simple string, or a colon and an optional smooth factor, although this second option is pointless. Even if a smooth factor is provided here, it won't be used. As an example, lets define a background counts column relative to the counts column:
- ((bcounts=(counts:3 − 5))
- This indicates that a new column, bcounts, should be created in the output MTL file, whose value is the value of counts smoothed over three rows, minus 5.0. Now bcounts can be used in a limits check:
- ((0 < bcounts) && (bcounts <= 2))
- The two lines could even be combined:
- ((0 < bcounts=(counts:3 − 5)) && (bcounts <= 2))
- This file can also contain a 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.
- User Limit string: This string follows the same rules as listed above for the limits FITS file.

ALGORITHMS

- Application of Limits (FITS File and user limit):
- In pipeline applications, the file will be used. For users, the userlimit parameter may be of more use. If both are provided, both will be used. If neither are provided, the tool will exit with an error message. One or the other, at least, must be specified. Each row of the gti_limits column is read into the program and, along with the user specified limit string (unless NONE), parsed to determine what columns ought to be in the MTL file, and what new columns are being defined. It will attempt to open the existent columns in the input MTL file. If they cannot be found there, they will be searched for in the list of new columns, as the new columns can be referenced as a normal column once they have been defined. If the column(s) still cannot be found, then a warning will be issued, and that entire limit string will be thrown out. This is why it is not recommended that the user chain limits checks for multiple columns together with &&. While this is possible, it might mean that one check is thrown out if a second cannot have it's column requirements met. Once a subset of the input limits has been found that is satisfied by the input MTL files, the gti limits will be tested once, to ensure that there is not a type conflict, i.e. checking if a short column is != to a string value. Integer/real values will be converted, but if numbers and strings are mixed, a warning will be issued. If any of the limits fail the conflict check, the limit will be thrown out, and the process must be started again − the tool will go through the process defined in the previous paragraph, and make sure the remaining limits checks have all their column conditions met. Then the conflict check is performed again. This cycle is repeated until there are no limits left, or the conflict check passes. Once this is done, the tool will loop over the rows of the table, smoothing the values, and then checking the row against all of the limits that remain. The row of smoothed values and new columns will be written to the output file. If the smoothing parameter is set to yes, then the value in the columns' smooth parameter will be used to determine the size of the boxcar smooth (this is described further below). Only those columns which were found in the limits FITS file will be included in the output MTL file, in addition to any new columns defined by the limits file. Smoothed values will be written to that file, if smoothing was allowed.
- Nature of the Boxcar Smooth:
- The limits FITS file defines the number of rows of data which a given column will be smoothed over. 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. The ends of MTL files require exceptions to these rules. Depending on the size of the smooth factor, the range of rows to comprise the smoothing will extend beyond the ends of the file. In these cases, the task truncates the range of rows, and averages by the number of rows actually used. Thus, if the first row is to be smoothed by 3, that would imply a range of −1 to 2. This is truncated to 1−2, and the averaging value is 2 rather than 3. Of course, it should be noted that the smoothing is done on a column by column basis − each column within a row has it's own individual smoothing factor. As each row is checked against the limits, each column is smoothed by the pertinent factor.

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.
- for the time being, it is best to use ()'s to make complicated limits explicit.

Example

```
mtl_build_gti gti_mtl.fits out_gti.fits out_flt_mtl.fits smooth=yes
clobber=yes
```

Parameters

name	type	ftype	def	reqd
infile	file	input		yes
outfile	file	output		yes
mtlfile	file	output		yes
userlimit	string	input		
lkupfile	file	input		
smooth	boolean		yes	
clobber	boolean		no	
verbose	integer		0	

Detailed Parameter Descriptions

Parameter=infile (file required filetype=input)

This is the input MTL files.

This is a FITS table with a BINTABLE first extension, which contains the MTL data. The file should contain the standard ASC header as well. The file's first column will be time.

Parameter=outfile (file required filetype=output)

Output GTI file

File to which the GTI will be written. An existing file may be specified, in which case the GTI information in that file is updated.

Parameter=mtlfile (file required filetype=output)

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 filetype=input)

Optional user defined limit string

This is a user input string that defines an (addtional) limit which the MTL data is checked against in creating

the GTI.

Parameter=lkupfile (file filetype=input)

Lookup table defining which MTL columns to check against

This is the lookup table defining which MTL columns to check against. The data formats section describes what sort of limit conditions are allowable, the algorithm section how the limits are used. The file can also contain an optional keyword, GTICLASS, which determines the value of the HDUCLAS2 keyword to write to the output GTI file.

Parameter=smooth (boolean default=yes)

Smooth the input MTL data?

This smoothing is not done in place. The input is unchanged. The smoothing is done on a copy and written to the output. If this is set to no, all smoothing factors in the limits file are overriden and set to one.

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.

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.

IRAF QPOE Files

Support for IRAF QPOE files has been removed from CIAO.

Bugs

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

See Also

chandra

mtl

tools

dmgti, dmimgcalc, dmtcalc, syntax

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