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

AHELP for CIAO 3.4

dmstat

Context: tools

Jump to: Description Examples Parameters CHANGES IN CIAO 3.3 CHANGES IN CIAO 3.2 CHANGES IN CIAO 3.0 USING VIRTUAL COLUMNS HANDLING 'INVALID' DATA ACCESSING STATISTICS FROM THE PARAMETER FILE Bugs See Also

Synopsis

Compute statistics for images and columns in tables.

Syntax

```
dmstat  infile [centroid] [median] [sigma] [clip] [nsigma] [maxiter]
[out_columns] [out_min] [out_min_loc] [out_max] [out_max_loc]
[out_mean] [out_median] [out_sigma] [out_sum] [out_good] [out_null]
[out_cnvrgd] [out_cntrd_log] [out_cntrd_phys] [out_sigma_cntrd]
```

Description

The dmstat tool calculates statistics of both columns in tables and images. The calculated values are printed on the screen and also stored in the parameter file, which allows easy access to the results from scripts.

Tables

When the input file is a table, the default option is to compute the mean, standard deviation, total, minimum, maximum, and the number of good and null rows for each of the columns in the table. Additional statistics include the median and the ability to perform an iterative, sigma−clipping algorithm to filter the data.

```
unix% dmstat "evt2.fits[cols x]"
x[pixel]
    min:        2479.8139648          @:        102340
    max:        6018.9042969          @:        126933
   mean:        4059.8933467
  sigma:        869.49332738
    sum:        765618747.21
   good:        188581
   null:        0
```

Here we use dmstat to calculate a number of statistics on the "x" column in the table "evt2.fits". If we had not used a "Data Model" filter to select the x column (see 'ahelp dmcols') then the statistics for each column in the table would have been calculated and printed to the screen. The values can also be recovered from the parameter

file:

```
unix% pget dmstat out_mean
4059.8933467
unix% set min = `pget dmstat out_min`
```

The screen output lists the minimum, maximum, mean, and standard deviation (here called 'sigma') values of the column. The "sum" field refers to the sum of the values in the column, and the "good" and "null" values refer to the number of valid and non−valid rows respectively (the handling of Null/NaN values is discussed below). The sum of the good and null columns gives the number of rows in the table (which equates to the number of events in an event file). Also listed, on the min and max lines after the "@" symbol, are the row numbers which correspond to the location of that value (or the first occurrence if there is more than one match).

The mean, sigma, and sum values are calculated using:

```
sum = sumof( x_i )
```

```
mean = sum / good
```

```
sigma = sqrt( sumof( (x_i − mean)^2 ) / good )
```

where x_i refers to the value of a single row. Note the use of good rather than (good−1) in the calculation of sigma.

Images

Two calculation modes are available for images: the centroid of the distribution (the default) or the mean level. Minimum and maximum values, the total signal, and the number of good and null values are also reported. At present dmstat can only work with two−dimensional images.

The output from the two modes is shown below for a 2D image:

```
unix% dmstat img.fits centroid=yes
EVENTS_IMAGE(x, y)
    min:        0            @:        ( 3722 3038 )
    max:        19           @:        ( 4205 3755 )
cntrd[log] :    ( 591.44825223 648.85355266 )
cntrd[phys]:    ( 4312.4482522 3685.8535527 )
sigma_cntrd:    ( 49.11625588 52.963861712 )
   good:        1523984
   null:        0
unix% pget dmstat out_cntrd_phys
4312.4482522,3685.8535527
```

and

```
unix% dmstat img.fits centroid=no
EVENTS_IMAGE
    min:        0            @:        ( 3722 3038 )
    max:        19           @:        ( 4205 3755 )
   mean:        0.0057441547943
  sigma:        0.084794915133
    sum:        8754
   good:        1523984
   null:        0
unix% pget dmstat out_sum
```

```
  8754
```

In both modes, the value and location − in physical coordinates − of the minimum and maximum pixel values is reported. Since there may be multiple pixels with either the minimum or maximum value, only the location of the first such pixel is returned. The good and null items refer to the number of pixels that are valid and invalid (i.e. pixels whose value equals the Null/NaN value) respectively.

For the centroid=yes case, the location of the centroid of the pixel distribution is reported in both logical and physical coordinate systems. For a set of N valid pixels with logical coordinates (x_i,y_j) and values f(x_i,y_j), the centroid (c_x,c_y) is calculated in the logical coordinate system via

```
psum = sumof( f(x_i,y_j) )
```

```
c_x = sumof( x_i * f(x_i,y_j) ) / psum
```

```
c_y = sumof( y_j * f(x_i,y_j) ) / psum
```

where the sums are over both i and j, and ignore those pixels equal to the Null/Nan value. The position in the physical coordinate system is then found by transforming (c_x,c_y). The "sigma centroid" values (sc_x,sc_y) are calculated using

```
sc_x = sqrt( (sumof ( abs(x_i) * (f(x_i,y_j)−mean)^2 )) / ngood)
```

```
sc_y = sqrt( (sumof ( abs(y_i) * (f(x_i,y_j)−mean)^2 )) / ngood)
```

As the centroid is calculated using all the valid pixels in the input image, the reported value will be affected by the presence of sources in the image other than the object of interest. It is therefore suggested that the image be filtered (see "ahelp dmimfiltering") to restrict the area to just the interesting source. Note that the centroid calculation is not well−defined if the image contains negative pixels.

For the centroid=no case, the mean, sigma, and sum values are calculated as:

```
sum = sumof( f(x_i,y_j) )
```

```
mean = sum / good
```

```
sigma = sqrt( sumof( (f(x_i,y_j) − mean)^2 ) / good )
```

Note the use of good rather than (good−1) in the calculation of sigma.

Sigma−clipping algorithm

An iterative, sigma−clipping algorithm can be used to reduce the effect of outliers on the computed statistics. Options exist to choose which statistics to compute if the execution time is a factor. If clip=yes with a table, then the algorithm works on each selected columns individually; for images, the setting is currently ignored if centroid=yes.

The sigma−clipping algorithm calculates the mean and standard deviation of the data, then removes all points which are more than nsigma times the standard deviation away from the mean. The procedure is repeated until either no more points are rejected (cnvrgd is set to Y) or the maximum number of iterations (as specified by the maxiter parameter) is reached (cnvrgd is set to N). The null value reports the total number of rejected elements,

i.e. both those rejected by the clipping algorithm and those whose values are Null (or NaN).

For example:

```
unix% dmstat "img.fits[sky=circle(4237,3694,40)]" centroid=no
EVENTS_IMAGE
    min:        0             @:        ( 4237 3654 )
    max:        4             @:        ( 4242 3690 )
   mean:        0.13990049751
  sigma:        0.38752947575
    sum:        703
   good:        5025
   null:        1536
```

and

```
unix% dmstat "img.fits[sky=circle(4237,3694,40)]" centroid=no clip=yes
EVENTS_IMAGE
    min:        0             @:        ( 4237 3654 )
    max:        1             @:        ( 4245 3655 )
   mean:        0.11377849506
  sigma:        0.31754204307
    sum:        564
   good:        4957
   null:        1604
 cnvrgd:        Y
unix% pget dmstat out_cnvrgd
Y
```

Since a Y is reported for cnvrgd, the number of rejected points (1604 − 1536 = 68) converged before the maximum number of iterations was reached. Note that the maximum pixel − and hence its location − has also changed.

Example 1

```
dmstat "intable.fits[cols energy]"
set mean = `pget dmstat out_mean`
```

Computes the standard set of statistics for the ENERGY column of intable.fits. The second line sets the variable "mean" to the mean value calculated for the column (using csh/tcsh syntax).

Example 2

```
dmstat "intable.fits[cols time,pha,pi]" median+
```

Computes the statistics for the TIME, PHA, and PI columns in the file intable.fits. The median value of each column will also be calculated and reported.

Example 3

```
dmstat "intable.fits[sky=circle(4096,4096,20),pha=:100][cols time]"
```

You can use the on−the−fly filtering capabilities of the Data Model (see "ahelp dm") to determine exactly what section of the data you want dmstat to consider. This example applies two filters to table:

- a spatial filter on the sky column
- restricts rows to those with a pha value less than 100

and then calculates the statistics of the time column of the remaining rows.

Example 4

```
dmstat intable.fits
```

Computes the statistics for all columns in the first interesting block of intable.fits.

Example 5

```
dmstat intable.fits median=no sigma=no
```

Computes the statistics, except for the median and standard deviation, for all columns in the first interesting block of intable.fits.

Example 6

```
dmstat inimage.fits
```

Computes the statistics for an image, reporting the position of the centroid of the distribution.

Example 7

```
dmstat inimage.fits centroid=no
```

Computes the mean level in the image with associated statistics.

Example 8

```
dmstat "inimage.fits[exclude sky=circle(500,500,20)]" centroid=no
```

Computes the statistics for an image after applying a spatial filter which excludes all pixels that lie within a circle of radius 20 with center (500,500) in sky coordinates. Those pixels excluded by the spatial filter are ignored in the calculation, only contributing to the number of excluded pixels listed in the "null" column.

See "ahelp dmimfiltering" for more information on filtering images.

Example 9

```
dmstat inimage.fits centroid=no clip=yes nsigma=2.5 maxiter=10
```

Computes the mean level in an image, with associated statistics, after removing all points which are more than 2.5 standard deviations away from the mean. This is repeated until either no more values are rejected or 10 iterations have been performed. If centroid=yes, then no clipping of the data would have occurred.

Example 10

```
dmstat "inimage.fits[sky=region(src.reg)]" centroid=no clip=yes
nsigma=2.5 maxiter=10
```

As with the previous example, but this time the calculation is restricted to only those pixels that lie within the region defined by the file "src.reg".

Example 11

```
dmstat intable.fits sigma- > /dev/null
set cols = `pget dmstat out_columns`
set means = `pget dmstat out_mean`
@ n = 1
@ nmax = `stk_count $cols echo+`
while ( $n <= $nmax )
set col = `stk_read_num $cols $n echo+`
set mean = `stk_read_num $means $n echo+`
printf "%-10s mean= %g\n" $col $mean
@ n++
end
```

Here we calculate the statistics for all the columns in intable.fits and then use the stack tools ("ahelp stack" and "ahelp tools /stk_/") to find the name and mean value for all columns in the file. This example uses csh/tcsh syntax.

Example 12

```
dmstat "*.fits" centroid=no
pget dmstat out_mean
```

Here we run dmstat on all files in the current directory that end in ".fits". Although the statistics for each entry in the stack will be written out to the screen, only those for the last file processed will be written to the parameter file.

Parameters

name	type	ftype	def	min	reqd	stacks
infile	file	input			yes	yes
centroid	boolean		yes			

median	boolean		no			
sigma	boolean		yes			
clip	boolean		no			
nsigma	real		3			
maxiter	integer		20	1		
out_columns	string					
out_min	string					
out_min_loc	string					
out_max	string					
out_max_loc	string					
out_mean	string					
out_median	string					
out_sigma	string					
out_sum	string					
out_good	string					
out_null	string					
out_cnvrgd	string					
out_cntrd_log	string					
out_cntrd_phys	string					
out_sigma_cntrd	string					

Detailed Parameter Descriptions

Parameter=infile (file required filetype=input stacks=yes)

Input stack of files

The stack of files on which statistics will be computed.

Parameter=centroid (boolean default=yes)

Calculate centroid of image?

This option is only used if the input is a two−dimensional image. If "yes", calculate the centroid of the distribution; otherwise, calculate the mean value.

Parameter=median (boolean default=no)

Calculate median value?

If set, calculate the median value of the distribution. Setting to "no" will speed up the execution time of the

program. The median is not calculated if the file is an image and centroid=yes.

Parameter=sigma (boolean default=yes)

Calculate the population standard deviation?

If set, calculate the standard deviation of the distribution. Setting to "no" will speed up the execution time of the program.

Parameter=clip (boolean default=no)

Calculate stats using sigma clipping?

Set to "yes" to use an iterative sigma−clipping algorithm; see also the parameters nsigma and maxiter. This parameter is ignored if the input file is an image and the centroid parameter is set to "yes."

Parameter=nsigma (real default=3)

Number of sigma to clip

If the clip parameter is set to "yes", values are removed if they lie more than nsigma standard deviations away from the mean.

Parameter=maxiter (integer default=20 min=1)

Maximum number of iterations

If the clip parameter is set to "yes", repeat the clipping algorithm until either no more points are removed − when the cnvrgd value will be reported as Y − or maxiter iterations has occurred, in which case the cnvrgd value will be reported as N.

Parameter=out_columns (string)

The names of the columns or image block.

A comma−separated list of column names when infile is a table or an image when centroid=yes, otherwise the name of the block containing the image (centroid=no). This parameter can be used to identify which element of the other out_* parameters belongs to which column.

Vector columns are listed as the individual components, and array columns are written out as a list of "name[index]" values, where index starts from 0. So, if infile="evt2.fits[cols time,sky,phas]" for a Chandra event file, then out_columns will be set to
"time,x,y,phas[0],phas[1],phas[2],phas[3],phas[4],phas[5],phas[6],phas[7],phas[8]".

This is an output parameter.

Parameter=out_min (string)

Minimum values.

The minimum values, stored as a comma−separated string.

This is an output parameter.

Parameter=out_min_loc (string)

Location of the minimum values.

The location of the minimum values, stored as a comma−separated string. For columns, the row number is used (starting from 1). For images, the physical coordinate system is used, when available; otherwise, the logical coordinate system is used.

This is an output parameter.

Parameter=out_max (string)

Maximum values.

The maximum values, stored as a comma−separated string.

This is an output parameter.

Parameter=out_max_loc (string)

The location of the maximum values.

The location of the minimum values, stored as a comma−separated string. For columns, the row number is used (starting from 1). For images, the physical coordinate system is used, when available; otherwise, the logical coordinate system is used.

This is an output parameter.

Parameter=out_mean (string)

The mean values.

The mean values stored as a comma−separated string. This parameter is set to "" when infile is an image and centroid=yes.

This is an output parameter.

Parameter=out_median (string)

The median values.

The median values stored as a comma−separated string. This parameter is set to "" when the median value has not been calculated.

This is an output parameter.

Parameter=out_sigma (string)

The standard deviation.

The standard deviation values stored as a comma−separated string. This parameter is set to "" when the standard deviation is not calculated.

This is an output parameter.

Parameter=out_sum (string)

The sum of the values.

The sum of all the values in a column, or an image, stored in a comma−separated list. This parameter is set to "" when infile is an image and centroid=yes.

This is an output parameter.

Parameter=out_good (string)

The number of good values.

The number of "good" values in the column or image, stored as a comma−separated list.

This is an output parameter.

Parameter=out_null (string)

The number of ignored values.

The number of ignored rows or pixels, stored as a comma−separated list.

This is an output parameter.

Parameter=out_cnvrgd (string)

Did the sigma clipping converge?

A Y value (yes) indicates that the sigma clipping converged, otherwise N (no) is used for each column or image. This parameter is set to "" when the sigma−clipping algorithm is not used.

This is an output parameter.

Parameter=out_cntrd_log (string)

The centroid of the image, in logical coordinates.

The centroid of the image in logical coordinates, stored as a comma−separated list. If the input is not an image or centroid=no, then this parameter is set to "".

This is an output parameter.

Parameter=out_cntrd_phys (string)

The centroid of the image, in physical coordinates.

The centroid of the image in physical coordinates, stored as a comma−separated list. If the input is not an image or centroid=no, then this parameter is set to "".

This is an output parameter.

Parameter=out_sigma_cntrd (string)

The standard deviation of the centroid, in logical coordinates.

The "standard deviation" of the image centroid in logical coordinates, stored as a comma−separated list. If the input is not an image or centroid=no, then this parameter is set to "".

This is an output parameter.

CHANGES IN CIAO 3.3

The statistics calculated by dmstat are now written to the parameter file as well as the screen. The parameter tools, such as pget, can be used to access these values. This means that it is no longer necessary to try to extract the calculated values from the screen output. For instance:

```
unix% set mean = `pget dmstat out_mean`
unix% set max = `pget dmstat out_max`
unix% set maxloc = `pget dmstat out_max_loc`
```

get the mean, max value, and location of the max value from the parameter file. Further information is given in the "ACCESSING STATISTICS FROM THE PARAMETER FILE" section below.

CHANGES IN CIAO 3.2

Minimum and Maximum location for vector and array columns

The row numbers of the minimum and maximum value of a column are now correctly reported for vector columns (such as "chip", "sky" and "eqpos") and array columns (such as "phas" in a level 1 event file). For example:

```
unix% dmstat "evt2.fits[cols chip]"
chip(chipx, chipy)[pixel]
    min:        ( 2 2 )       @:        ( 959 489 )
    max:        ( 1023 1023 )         @:        ( 73 212 )
...
```

and

```
unix% dmstat "evt1.fits[cols phas]"
phas[adu]
    min:        [ −3477 −3329 −3476 −3476 39 −3475 −3478 −3426 −3477 ]   @:  [ 1689380 324809 8547
    max:        [ 3373 1861 3434 1866 3749 1871 3578 1856 3224 ]         @:  [ 500028 1401773 7069
...
```

Image support is limited to two−dimensional data

The support for images with more than two dimensions has been removed in this release.

CHANGES IN CIAO 3.0

Improved handling of spatial filters

Normally when a spatial filter is applied to an image using the Data Model (see "ahelp dmimfiltering"), pixels excluded by the filter are set to 0. Prior to CIAO 3.0, the only way to get dmstat to ignore such filtered pixels was to set them to the null/NaN value for the image (see "ahelp dmopt"). This is no longer needed since dmstat now uses only those pixels that lie within the data subspace (see "ahelp subspace") of the image.

In CIAO 3.0 one can therefore say:

```
unix% dmstat "img.fits[sky=circle(100,100,5)]"
```

whereas you used to have to use something like:

```
unix% dmstat "img.fits[sky=circle(100,100,5)][opt null=−999]"
```

This new behavior is particularly useful since it recognizes spatial filters that have previously been applied to the data (assuming the coordinate system matches). For instance, after the following set of commands:

```
unix% dmcopy "evt.fits[sky=region(source.reg)]" evt.source
unix% dmcopy "evt.source[bin sky=1]" img.source
unix% dmstat img.source
```

the calculation will ignore those pixels in img.source that are not within the region defined by the file source.reg.

Output precision

The maximum number of significant digits used to display floating−point numbers has been increased from 9 to 11. For example, dmstat will now display 0.0057441547943 rather than 0.00574415479.

Minimum and Maximum row location for tables

The row number of the the minimum and maximum value of a column is now included in the output to match the behavior with images. For example:

```
unix% dmstat "evt2.fits[cols x]"
x[pixel]
    min:        2479.8139648          @:        102340
    max:        6018.9042969          @:        126933
...
```

USING VIRTUAL COLUMNS

If the table contains a column with a World Coordinate System attached to it − for instance, the SKY column of a Chandra event list − then dmstat can calculate statistics in both the original and transformed coordinate frames. All you have to do is include the names of the columns you require − e.g. "evt2.fits[cols ra]" or "evt2.fits[cols eqpos]" − as part of the infile parameter. For example:

```
unix% dmstat "evt2.fits[cols eqpos]"
EQPOS(RA, DEC)[deg]
    min:        ( 258.57348385 67.020579406 )        @:        ( 31812 29626 )
    max:        ( 259.82637267 67.421050984 )        @:        ( 206 7825 )
   mean:        ( 259.26622974 67.217694774 )
  sigma:        ( 0.30743608783 0.094642855434 )
    sum:        ( 48892684.871 12675980.098 )
   good:        ( 188581 188581 )
   null:        ( 0 0 )
```

HANDLING 'INVALID' DATA

Tables and images can include elements which do not contain valid data; for instance, the result of a division by 0 or an event in a grating observation which does not have a grating order. Such items are generally flagged by a "special" number (see the "null" option in "ahelp dmopt") and dmstat will ignore them during the calculation. As part of its output, dmstat reports the number of valid (good) and invalid (null) rows/pixels it processed.

Pixels in images can also be ignored if they lie outside the region of interest − i.e. a spatial filter which has been applied to the image. Prior to CIAO 3.0, you had to explicitly set these pixels to a "null" value using the "opt null" option of the Data Model; this is no longer necessary since dmstat uses the subspace information of the image (see "ahelp subspace") to find out which pixels to exclude from the calculation.

ACCESSING STATISTICS FROM THE PARAMETER FILE

New to CIAO 3.3 is the ability of dmstat to write the calculated values to the parameter file (using the "out_*" parameters). This means that you no longer need to parse the screen output to access these values. The following csh/tcsh expression will set the variable m to the mean value calculated by dmstat.

```
unix% set m = `pget dmstat out_mean`
```

Not all the fields are written to the parameter file; the output depends on what options are selected and the type of the input file. For example, out_median is only filled in if the median option is set to true and either the input file is not an image or the centroid option is set to false. The contents of some of the fields also depends on the options chosen: e.g. when analyzing an image, the out_columns parameter is set to the block name when the centroid option is set to no, but it is set to a comma−separated list of the names of the components of the physical coordinate system when centroid is set to "yes".

Using multiple columns

When given a scalar column, such as "evt2.fits[cols x]", the output values will also be scalar values. For more complicated situations, such as "evt2.fits[cols sky]" and "evt2.fits[cols time,energy]", the output values will be written out as a comma−separated list. As these values can be treaded as stacks (see "ahelp stack"), the stack tools and module ("ahelp /stk_/") can be used to access individual elements of the list. The contents of the out_columns parameter can be used to map between position and column. So, after

```
unix% dmstat "evt2.fits[cols time,sky]" sigma−
time[s]
    min:        68010810.424          @:        1
    max:        68010862.403          @:        1973
   mean:        68010837.09
    sum:        1.3602167418e+11
   good:        2000
   null:        0
```

```
sky(x, y)[pixel]
    min:        ( 2495.1342773 2870.4226074 )        @:        ( 206 940 )
    max:        ( 5968.3662109 5709.3413086 )        @:        ( 1610 1384 )
   mean:        ( 3969.3936443 4225.1935254 )
    sum:        ( 7938787.2886 8450387.0508 )
   good:        ( 2000 2000 )
   null:        ( 0 0 )
```

we can say

```
unix% set cols = `pget dmstat out_columns`
unix% echo $cols
time,x,y
unix% set means = `pget dmstat out_mean`
unix% echo $means
68010837.09,3969.3936443,4225.1935254
unix% stk_where $cols x echo+
2
unix% stk_read_num $means 2 echo+
3969.3936443
```

Note that although only two columns − namely "time" and "sky" − were given to dmstat, the output contains three values, since the "sky" column has been broken up into its constituent columns ("x" and "y"). Columns that are unsupported by dmstat − i.e. bit arrays and character strings − are not written out to the parameter file.

For images, the number of values written out to each parameter depends on the dimensionality of the image (although dmstat currently only accepts two−dimensional images).

Setting the infile parameter to a stack

When the infile parameter is a stack − such as "img1.fits,img2.fits" or "@file.lis" − only the statistics for the last item in the stack are written to the parameter file. The filename can be found from the parameter file using something like the following:

```
unix% set files = `pget dmstat infile`
unix% set n = `stk_count "$files" echo+`
unix% set file = `stk_read_num "$files" $n echo+`
```

Bugs

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

See Also

tools

dmcopy, dmextract, dmgroupreg, dmlist

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