



AHELP for CIAO 3.4

write

Context: [sherpa](#)

Jump to: [Description](#) [Examples](#) [Bugs](#) [See Also](#)

Synopsis

Causes the specified information to be written to the screen or to a file.

Syntax

```
sherpa> WRITE <arg> [# [ID]] [<filename> [<filetype>]]
```

specifies the number of the dataset (default dataset number is 1). The ID modifier is used for writing background datasets, and then if and only if the Sherpa state object variable multiback is set to 1, i.e., if more than one background dataset is to be associated with a single source dataset. The ID modifier may be any unreserved string (e.g., A, foo, etc.), i.e., a string that is not a parsable command.

The name of the file to which data is written is <filename>. If <filename> is not specified, information is written to the screen.

Description

The argument <arg_n> may be any of the following:

WRITE Command Arguments

| Argument | Displays |
|---|--|
| {DATA DCOUNTS} | The source data values |
| {BACK BDCOUNTS} | The background data values |
| ERRORS | The errors associated with source data points |
| BERRORS | The errors associated with background data points |
| {SYSERRORS BSYSERRORS} | The assigned systematic errors for the source background data values |
| {STATERRORS BSTATERRORS} | The estimated statistical errors for the source background data values |
| {{MODEL MCOUNTS} {BMODEL BMCOUNTS}} | The (convolved) source background model amplitudes |
| {DELCHI BDELCHI} | The sigma residuals of the source background fit: (data – model)/error |
| {RESIDUALS BRESIDUALS} | The absolute residuals of the source background fit: (data – model) |
| {RATIO BRATIO} | The ratio (data/model) for source background |
| {CHI SQU BCHISQU} | The contributions to the chi–square statistic from each source background data point |

| | |
|--------------------------|---|
| {STATISTIC BSTATISTIC} | The contributions to the current statistic |
| {WEIGHT BWEIGHT} | The statistic weight value assigned to each source background data point |
| {FILTER BFILTER} | The mask value (0 1) for each source background data point |
| {GROUP BGROUP} | The grouping value (1 -1) associated with each source background data point |
| {QUALITY BQUALITY} | The quality value (0 2 5) associated with each source background data point |
| <sherpa_model_stack> | The (unconvolved) model amplitudes for the specified model stack (SOURCE, {BACKGROUND BG}, [B]NOISE, or KERNEL) |
| <model_stack> | The (unconvolved) model amplitudes for the specified user-defined model stack |
| <sherpa_modelname> | The (unconvolved) amplitudes of the specified model component (e.g., GAUSS) |
| <modelname> | The (unconvolved) amplitudes of the specified model component (e.g., g) |
| {ARF BARF} | The unfiltered source background ARF file contents |
| {EXPMAP BEXPMAP} | The unfiltered source background exposure map file contents |
| {PSF BPSF} | The unfiltered source background PSF file contents |
| MDL | A model descriptor list file. |

The argument <filetype> can specify the desired format for the file that is written:

WRITE File Type Arguments

| <filetype> | Description |
|------------|-----------------------|
| ASCII | 1-D ASCII |
| FITS | 2-D FITS image |
| FITSIMAGE | 2-D FITS image |
| FITSBIN | 1-D FITS binary table |
| PHA | PHA file |

Note the following:

- if <filetype> is not specified, the default output file format is ASCII for 1-D data and FITS for 2-D data.
- the WRITE command utilizes the settings of ANALYSIS and PLOTY to determine the units of the output. An exception to this rule in CIAO 3.0 is the writing out of data to PHA format files; here, counts are written out regardless of the setting of PLOTY.
- if the data are grouped, then the channels in the file produced by WRITE DATA ... PHA will represent the grouped channels. However, since no grouping information is written to the file, the bin energies will be incorrect. Therefore, WRITE ... PHA should be used only with ungrouped data.
- the VARMM/S-Lang module functions writeascii and writefits, which may be used within Sherpa, provide alternate means of writing data to disk.

The WRITE command may be issued at any time. Each time the command is issued the specified file <filename> is overwritten, but only if the state variable sherpa.clobber is set to 1. Otherwise, an error message is displayed if the file exists.

Using Data Model Filters

This command is able to take any Data Model virtual file specification (see "ahelp dmsyntax"). If you can do

```
unix% dmcopu "infile.fits[spec 1][spec 2]" outfile.fits
```

you can also do

```
sherpa> write "infile.fits[spec 1][spec 2]"
```

This is especially useful when working with very large files. For example:

```
sherpa> write "evt.fits[bin sky=4][opt mem=100]"
```

bins the event file by a factor of four and allocates additional memory.

Example 1

Write an ASCII dataset to a backup ASCII file:

```
sherpa> READ DATA 2 example.dat 1 2
sherpa> WRITE DATA 2 example_backup.dat
Write X-Axis: Bin Y-Axis: Flux (Counts)
sherpa> WRITE DATA 2 example_backup.dat
Write X-Axis: Bin Y-Axis: Flux (Counts)
Error: file exists and sherpa.clobber = 0.
sherpa> sherpa.clobber = 1
sherpa> WRITE DATA 2 example_backup.dat
Write X-Axis: Bin Y-Axis: Flux (Counts)
```

The second command writes dataset number 2 to an ASCII file named example_backup.dat. This file cannot be overwritten unless the state variable sherpa.clobber is set to 1.

Example 2

Write a PHA dataset to a backup ASCII file:

```
sherpa> READ DATA example.pha
The inferred file type is PHA. If this is not what you want, please
specify the type explicitly in the data command.
WARNING: using systematic errors specified in the PHA file.
RMF is being input from:
 <directory_path>/example.rmf
ARF is being input from:
 <directory_path>/example.arf
Background data are being input from:
 <directory_path>/example_bkg.pha
sherpa> WRITE DATA pha.dat ASCII
Write X-Axis: Energy (keV) Y-Axis: Flux (Counts/sec/keV)
```

Note that the ASCII argument in the command WRITE DATA pha.dat ASCII is not actually needed, since the default for 1-D data is ASCII format.

Example 3

Extract three columns from a FITSBIN file and write them to an ASCII file:

```
sherpa> READ DATA "example1_bin.fits[cols x1,x2,x3]" FITSBIN
sherpa> WRITE DATA threecols.txt ASCII 1 2 3
```

```
Write X-Axes: (Bin,Bin) Y-Axis: Counts
```

Example 4

Make a fit to data and save the best-fit amplitudes for one of the model components, along with information about the fit:

```
sherpa> ERASE ALL
sherpa> READ DATA example1.dat
sherpa> PARAMPROMPT OFF
Model parameter prompting is off
sherpa> SOURCE = POLY[p1] + POW[p2]
sherpa> FIT
LVMQT: V2.0
LVMQT: initial statistic value = 3070.54
LVMQT: final statistic value = 0.252662 at iteration 14
      p1.c0  0.256496
      p2.gamma -2.03063
      p2.ampl 15.6881

sherpa> WRITE p2 powlaw.mod ASCII
Write X-Axis: Bin Y-Axis: Flux (Photons/bin)
sherpa> WRITE SOURCE source.mod
Write X-Axis: Bin Y-Axis: Flux (Photons/bin)
sherpa> var1 = get_axes(1)
sherpa> print(var1)
axistype      = Channels
axisunits     = unknown
lo            = NULL
hi            = NULL
mid           = Float_Type[8]
sherpa> var2 = get_data(1)
sherpa> var3 = get_errors(1)
sherpa> var4 = get_mcounts(1)
sherpa> writeascii("fit.dat", var1.mid, var2, var3, var4)
```

The last command writes out a 4-column file with x-axis values and associated data, errors, and predicted model counts.

Bugs

See the [Sherpa bug pages](#) online for an up-to-date listing of known bugs.

See Also

chandra

[guide](#)

sherpa

[get analysis](#), [get arf axes](#), [get axes](#), [get coord](#), [get data](#), [get energy axes](#), [get errors](#), [get filter](#), [get filter expr](#), [get fit](#), [get fluxed spectrum](#), [get ftest](#), [get metadata](#), [get photon axes](#), [get photon energy axes](#), [get photon wave axes](#), [get qvalue](#), [get raw axes](#), [get record](#), [get source](#), [get statistic](#), [get stats](#), [get syserrors](#), [get wave axes](#), [get weights](#), [record](#), [save](#)