Chandra X-ray Center

AHELP for CIAO 3.4   **set_axes**   Context: sherpa

Jump to: Description Examples Bugs See Also

Synopsis

Module functions for creating new source or background dataspaces.

Syntax

```
Integer_Type set_axes([Integer_Type,]Struct_Type|Array_Type)
Integer_Type set_baxes([Integer_Type,]Struct_Type|Array_Type)

Success/Error Return Values: 1/0

Arguments:

(1) Dataset number (default 1)

(2) A structure containing fields that define a 1-D dataspace, or

(2) An array that defines a 1-D dataspace, or

(2) An array of N structures containing fields that define an N-D
dataspace, or

(2) An array of N arrays that define an N-D dataspace.

Note that if only one argument is provided, it is assumed to be one of
the quantities listed for argument 2, and the dataset is assumed to be
dataset 1.
```

Description

The set_axes function is the equivalent of Sherpa's DATASPACE command. (The function set_baxes, used to set up dataspaces for background datasets, has no Sherpa command analogue.) A ``dataspace'' is an N–dimensional grid representing the independent variables of a dataset (e.g., the x values in y = f(x)).

A given axis may be defined using an array, or a structure; the former are appropriate for unbinned data entry, while structures are appropriate for either unbinned or binned data entry (the equivalent of DATASPACE HISTOGRAM). The structures must contain fields named lo and hi (for binned dataspaces), or mid (for unbinned dataspaces). These fields point to numeric arrays. (set_axes can thus process the structures output by get_axes. Note that the array lengths of the lo and hi fields of each input structure must be equal (though they may vary from axis to axis; e.g., to define a 512 x 1024 binned dataspace).

The set_axes functions should be used with care, as it instructs Sherpa to replace any dataset previously input

with the same dataset number, with the new dataspace.

See the related Sherpa command DATASPACE for more information.

Example 1

Create a new 1−D dataspace:

```
sherpa> DATA spec.dat
sherpa> str = get_axes(1)
sherpa> print(str)
axistype        =   Channels
axisunits       =   unknown
lo              =   NULL
hi              =   NULL
mid             =   Float_Type[100]
sherpa> theaxis = @(foo[0].mid)
sherpa> () = set_axes(1,theaxis)
```

In this example, the user takes a dataset and ends up replacing the read−in counts with zeroes. In the first step, unbinned ASCII data are read in. Because they are unbinned, get_axes returns a structure for which only the mid field is non−NULL. (There are no lower and upper bin boundaries.) The array pointed to by str.mid is then copied (the `@' sign denotes a total copy of array contents), and used to define a new dataspace via set_axes.

Example 2

Create a new binned 2−D dataspace:

```
sherpa> variable a0 = struct{lo,hi}
sherpa> a0.lo = [1:1024]
sherpa> a0.hi = [2:1025]
sherpa> variable a1 = struct{lo,hi}
sherpa> a1.lo = [1:1024]
sherpa> a1.hi = [2:1025]
sherpa> a = Struct_Type[2]
sherpa> a[0] = a0
sherpa> a[1] = a1
sherpa> set_axes(1,a)
1
```

In this example, the user creates two structures which are minimally sufficient for setting up a binned dataspace (such a structure must have fields lo and hi, and it could have more fields, but these are ignored). Arrays are defined for each of the structure fields, such that the first bin has lower and upper bin boundaries 1 and 2, the second bin 2 and 3, etc. An array of structures a is then defined, and sent to Sherpa via set_axes. The call returns 1, indicating that Sherpa successfully set up a 1024 x 1024 dataspace.

Bugs

See the Sherpa bug pages online for an up−to−date listing of known bugs.

See Also

chandra
        guide
sherpa

autoest, back, berrors, bsyserrors, bye, calc_kcorr, coord, data, dataspace, dcounts, dollarsign, echo, eflux, eqwidth, erase, fakeit, feffile, flux, get, get_dcounts_sum, get_dir, get_eflux, get_eqwidth, get_filename, get_flux2d, get_flux_str, get_lfactorial, get_mcounts_sum, get_pflux, get_source_components, get_verbose, group, groupbycounts, guess, is, is_subtracted, journal, list, list_par, load, load_arf, load_ascii, load_back_from, load_backset, load_dataset, load_fitsbin, load_image, load_inst, load_inst_from, load_pha, load_pha2, load_rmf, mcounts, numbersign, paramest, plot_eprof, plot_rprof, prompt, read, reset, run, set, set_analysis, set_backscale, set_coord, set_data, set_dataspace, set_dir, set_exptime, set_subtract, set_verbose, set_weights, setback, setdata, setplot, sherpa–module, sherpa_plotfns, sherpa_utils, show, simspec, subtract, ungroup, unsubtract, use, version

Example 1   3

Example 1