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

AHELP for CIAO 3.4

group

Context: modules

Synopsis

The S−Lang interface to the CXC grouping library

Description

The group module is the interface between the S−Lang interpreter (see "ahelp slang") and the CXC group library (see "ahelp dmgroup"), which bins histogram data based on various user−selected rules. This document provides an overview of the features of the group module, and shows how to use it in a S−Lang program. Detailed descriptions of each function are provided by individual ahelp pages.

The group module is not available by default; to use it in a S−Lang program, it must be loaded using the S−Lang require() function:

```
require("group");
```

Functions provided by the module

The following functions are provided by the module; use "ahelp <function>" to get a detailed description of a function:

Function name with arguments
(grouping, quality) = grpNumCounts(countsArray, numCounts [,maxLength,tabStops]);
(grouping, quality) = grpNumBins(numChans, numBins [, tabStops]);
(grouping, quality) = grpBinWidth(numChans, binWidth [, tabStops]);
(grouping, quality) = grpSnr(countsArray, snr [,maxLength, tabStops,errorCol]);
(grouping, quality) = grpAdaptive(countsArray, minCounts [,maxLength, tabStops]);
(grouping, quality) = grpAdaptiveSnr(countsArray, snr [,maxLength, tabStops, errorCol]);
(grouping, quality) = grpMinSlope(dataArray, binArray, slope [,maxLength, tabStops]);
(grouping, quality) = grpMaxSlope(dataArray, binArray, slope [,maxLength, tabStops]);
(grouping, quality) = grpBin(dataArray, binLowArray, binHighArray [,tabStops]);
(grouping, quality) = grpBinFile(dataArray, fDataArray, fGroupingCol, fQualCol [,tabStops]);
grpdata = grpGetGroupSum(dataArray, groupCol);
chanspergrp = grpGetChansPerGroup(groupCol);
grpnum = grpGetGrpNum(groupCol);

Purpose

The group module is intended to take an array of counts and group it, summing certain adjacent bins to increase the overall signal to noise. The available routines can group bins until each bin has a minimum number of counts (grpNumCounts), or until there are a fixed number of bins (grpNumBins), or simply group a fixed number of bins together (grpBinWidth). More complex grouping algorithms exist to bin data adaptively until a minimum signal to noise ratio is met, as well as other methods. See "ahelp <function>" for more details on the individual functions.

Optional Parameters

Some parameters are optional; these are shown in the function calls listed above inside []s. Any optional parameter can be omitted; if you want to use the default for one optional parameter but set an optional parameter that follows it, simply include an extra comma for each skipped parameter. See the EXAMPLE section below for more details.

Function output

Most of the grouping functions output two arrays, a grouping array and a quality array. The grouping array is a vector the same length as the data to be grouped. The grouping vector is 1 at the start of a new "grouped" bin and −1 at a continuation. A grouping array of

```
[ 1, −1, 1, −1, 1, −1 ]
```

indicates that a 6 element array will be grouped into 3 groups, each containing 2 elements. The quality array is the same length, with values of 0 to indicate good data, 5 to indicate data omitted by the user in the grouping call (i.e., within a tabStop), or 2 to indicate data without enough channels to make a complete bin (i.e., the 13th channel in a 13−element array binned by 3s). For the complete definition of these two arrays, see the OGIP Memo OGIP/92−007

Example

```
sherpa> require("group")
sherpa> ChanArr = [1:1024:1]
sherpa> CountArr = 5 + 3*sin(ChanArr/100.)
% Group to get at least 5 counts per bin, with a maximum of 10 channels
binned together.
sherpa> (grpArr1, QualArr1) = grpNumCounts(CountArr, 5, 10)
% Group input vector into 150 bins
sherpa> (grpArr2, QualArr2) = grpNumBins(1024, 150)
% Group every 8 bins together, but skip channels 100, 200, and 245.
sherpa> (grpArr3, QualArr3) = grpBinWidth(1024, 8, [100,200,245])
% Sum the grouped data, using grouping array grpArr1
sherpa> GrpCount1 = grpGetGroupSum(CountArr, grpArr1)
% Extract only 1 channel for each binned group
sherpa> GrpChan1 = ChanArr[where(grpArr1==1)]
```

EXAMPLE OF USING THE GROUP MODULE

The following routine can be used to group a PHA spectrum interactively, allowing it to be plotted easily.

```
 %
 % Usage:
 %  (channel, counts) = bin_pha_adaptive( "data.pha" , 15);
 %
```

```
require("varmm");
require("group");
define bin_pha_adaptive ( phafile, minCts ) {

   variable grouping, quality;
   variable pha = readfile(phafile);
   if (orelse {typeof(pha) != Struct_Type} {pha._filetype != PHA_I}) {
      vmessage("Error: %s not found or not a legal PHA file.",pha);
      return NULL;
   }
   (grouping, quality) = grpNumCounts(pha.counts, minCts);
   variable grpData = grpGetGroupSum(pha.counts, grouping);
   variable grpChan = grpGetGrpNum(grouping);

   grpData = grpData[where(grouping==1 and quality==0)];
   grpChan = grpChan[where(grouping==1 and quality==0)];
   return grpChan, grpData;
} % bin_pha_adaptive ( phafile, minCts )
```

With the above function, one can see what effect a particular value for minCts will have on the plotted spectrum.

```
sherpa> evalfile("bin_pha_adaptive.sl");
1
sherpa> (c,d) = bin_pha_adaptive("ngc892.pha",15)
sherpa> curve(c,d)
sherpa> redraw
```

CHANGES IN CIAO 3.2

The module can now be loaded by using the

```
require("group");
```

statement, although the previous method (loading with the import command) still works.

Bugs

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

See Also

group

grpadaptive, grpadaptivesnr, grpbin, grpbinfile, grpbinwidth, grpgetchanspergroup, grpgetgroupsum, grpgetgrpnum, grpmaxslope, grpminslope, grpnumbins, grpnumcounts, grpsnr

---

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