Subject: Re: HIST_ND
Posted by JD Smith on Tue, 19 Aug 2003 16:16:59 GMT
View Forum Message <> Reply to Message

On Tue, 12 Aug 2003 09:32:51 -0700, matthew angling wrote:

> Hi!
>
> I'm trying to increment some voxel values based on xyz tuples - a
> classic use for a histogram. I'm trying to use JD's HIST_ND, but it's
> proving too much for my feeble brain! So before I get into the misteries
> of reverse indices, I've started to try to understand it by comparing
> output from HIST_ND with RSI's HIST_2D as per the following 2d example.
> Why do the two routines give different results?
>
> IDL> ind1=[0,1,1,2]
> IDL> ind2=[0,2,2,3]
> IDL> print,hist_2d(ind1,ind2)
> 1 0 0
> 0 0 0
> 0 2 0
> 0 0 1
> IDL> print,hist_nd(transpose([[ind1],[ind2]]),1)
> 1 0
> 0 0
> 0 3
>
> What am I missing here?

Nothing fundamental. This is analogous to the histogram on the razor's edge issue raised recently. You're constructing integer bins from min->max and then dropping integers which fall just on the boundary of those bins in. HIST_2D and HIST_ND do slightly different things in this case, either of which could be considered correct. If you'd like the same behavior as HIST_2D, you might change HIST_ND to read:

 endif else nbins=long((mx-mn)/bs+1)

I probably should have used this to begin with for consistency anyway, so perhaps I'll make this a permanent change. For large histograms, this doesn't really change much. Note that I'll stick with my notion of the NBINS keyword giving you exactly that many bins, since it makes much better sense than HISTOGRAM's NBINS treatment.

JD