Subject: Re: Help setting up an array
Posted by John-David T. Smith on Thu, 29 Mar 2001 03:56:12 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Peter Thorne <peter.thorne@uea.ac.uk> writes:
>
>> Thanks to everyone who has replied. At nearly 11pm I'm not sure whether
>> its exactly what I'm looking for, I shall investigate further tomorrow.
>> It seems like as suggested I have been looking at it for too long so
>> have tried to explain it in far too much difficulty, sorry. So, I'll
>> give a hopefully better example:
>>
>> 3-D (to keep everyone happy, theoretically could be expected to be 2 to
>> 5 dimensional)
>>
>> Locations array (points within a 3-D ellipsoid)
>> x y z (coordinates)
>> (1.5,3.4,2.0) point 0
>> (3.,-0.5,6.3) point 1
>> (1.3,2,.-4.5) point 2
>> (-0.1,1.7,0.1) point 3
>> .
>> .
>> .
>> .
>> (3.1,9.2,-1.4) point npoint
>>
>> npoint is of order (10,000)
>>
>> From this I wish to create a say 50x50x50 grid which covers all
>> plausible values (found by min and max in each column of the locations
>> array).
>>
>> Then I need to rebin each of these points into the 3-D grid-space, so
>> each grid-box has a value which is the number of these original points
>> which fall within that grid-box. Other considerations are peripheral,
>> the problem arises in this transformation from the locations array to a
>> finite difference grid in which the values can be rebinned and how they
>> are rebinned.
>>
>> This may have been covered already, but as my IDL license is at work and
>> not home I can't check :(
>>
>> Thanks again for all the pointers and comments
>
> Yeah, this is indeed a job for HISTOGRAM. What you need to do is

> check out how HIST_2D is implemented. This can generalize to many
> dimensions. The one thing that HIST_2D does *not* do is to pass along
> the REVERSE_INDICES array, but it is trivial to add this if you need
> it (say, if you want to "invert" the histogram and find out which
> points fall in a particular bin).
>
> The end result will be something like this (for 3d, but not tested):
>
> i = floor((x-xmin)/xstep)
> j = floor((y-ymin)/ystep)
> k = floor((z-zmin)/zstep)
>
> ijk = i + nx*(j + ny*k)
>
> h = histogram(ijk)
> h = reform(h, nx, ny, nz, /overwrite)
>
> If you are clever you can generalize this to multi-d. Doing
> multi-dimensional histograms, with weighting, is something I've been
> intending to do in a program called CMHISTOGRAM. Alas it is only half
> written.

OK, I took the bait, and wasted some time. It's amazing how productive
you can be when you have other things to do.

Attached you'll find HIST_ND, for n-dimensional histograms. It's well
documented, and pretty straightforward. Try it out with some random
data:

```
IDL> c=randomu(sd,3,100)
IDL> plot_3dbox,x[0,*],x[1,*],x[2,*],PSYM=4,CHARSIZE=1.5,$
 ZRANGE=[0,1],ZSTYLE=1
IDL> print,hist_nd(c,NBINS=[3,3,3],MIN=0,MAX=1)
```

Enjoy,

JD

```
;+
; NAME:
;  HIST_ND
;
; PURPOSE:
;
;      Perform an N-dimensional histogram, also known as the joint
;      density function of N variables, ala HIST_2D.
;
; CALLING SEQUENCE:
;  hist=HIST_ND(V,BINSIZE,MIN=MIN,MAX=MAX,NBINS=NBINS,REVERSE_I NDICES=ri)
```

```
;
; INPUTS:
;
; V: A NxP array representing P data points in N dimensions.
;
; BINSIZE: The size of the bin to use. Either a P point vector
;       specifying a separate size for each dimension, or a scalar,
;       which will be used for all dimensions.  If BINSIZE is not
;       passed, NBINS must be.
;
;
; OPTIONAL INPUTS:
;
;       MIN: The minimum value for the histogram.  Either a P point
;       vector specifying a separate minimum for each dimension, or a
;       scalar, which will be used for all dimensions.  If omitted,
;       the natural minimum within the dataset will be used.
;
;       MAX: The maximum value for the histogram.  Either a P point
;       vector specifying a separate maximmum for each dimension, or a
;       scalar, which will be used for all dimensions. If omitted, the
;       natural maximum within the dataset will be used.
;
;       NBINS: Rather than specifying the binsize, you can pass NBINS,
;       the number of bins in each dimension, which can be a P point
;       vector, or a scalar.  If BINSIZE it also passed, NBINS will be
;       ignored, otherwise BINSIZE will then be calculated as
;       binsize=(max-min)/nbins.  Note that *unlike* RSI's version of
;       histogram as of IDL 5.4, this keyword actually works as
;       advertised, giving you NBINS bins over the range min to max.
;
; KEYWORD PARAMETERS:
;
; MIN,MAX: See above
;
;       REVERSE_INDICES: Set to a named variable to receive the
;       reverse indices, for mapping which points occurred in a given
;       bin.
;
; OUTPUTS:
;
;       The N-Dimensional histogram, of size N1xN2xN3x...xND where the
;       Ni's are the number of bins implied by the data, and input
;       min, max and binsize.
;
; OPTIONAL OUTPUTS:
;
; The reverse indices
```

```
;
; EXAMPLE:
;
; v=randomu(sd,2,100)
; h=hist_2d(v,.25,MIN=0,MAX=1,REVERSE_INDICES=ri)
;
; MODIFICATION HISTORY:
;
;       Wed Mar 28 19:41:10 2001, JD Smith <jdsmith@astro.cornell.edu>
;
;  Written, based on HIST_2D, and suggestions of CM.
;
;-

function hist_nd,V,bs,MIN=mn,MAX=mx,NBINS=nb,REVERSE_INDICES=ri
  s=size(V,/DIMENSIONS)
  if n_elements(s) ne 2 then message,'Input must be N x P'

  if n_elements(mx) eq 0 then begin
     mx=make_array(s[0],TYPE=size(V,/TYPE))
     need_mn=n_elements(mn) eq 0
     if need_mn then mn=mx
     for i=0,s[0]-1 do begin
        mx[i]=max(V[i,*],MIN=tmn)
        if need_mn then mn[i]=tmn
     endfor
  endif

  if n_elements(mn) eq 1 and s[0] gt 1 then mn=replicate(mn,s[0])
  if n_elements(mx) eq 1 and s[0] gt 1 then mx=replicate(mx,s[0])
  if n_elements(bs) eq 1 and s[0] gt 1 then bs=replicate(bs,s[0])

  if n_elements(bs) eq 0 and n_elements(nb) ne 0 then bs=float(mx-mn)/nb else $
     message,'Must pass one of binsize or NBINS'
  nbins=long((mx-mn)/bs)

  tmx=nbins[s[0]-1]

  h=(nbins[s[0]-1]-1)<long((V[s[0]-1,*]-mn[s[0]-1])/bs[s[0]-1]) >0L
  for i=s[0]-2,0,-1 do begin
     h=nbins[i]*h+((nbins[i]-1)<long((V[i,*]-mn[i])/bs[i])>0L)
     tmx=tmx*nbins[i]
  endfor

  ret=make_array(TYPE=3,DIMENSION=nbins,/NOZERO)
  if arg_present(ri) then $
     ret[0]=histogram(h,min=0,max=tmx-1,REVERSE_INDICES=ri) $
  else $
```

```
    ret[0]=histogram(h,min=0,max=tmx-1)
  return,ret
end
```

File Attachments

1) hist_nd.pro, downloaded 148 times