Subject: Re: Help setting up an array
Posted by Pavel A. Romashkin on Wed, 28 Mar 2001 17:25:30 GMT
View Forum Message <> Reply to Message

Wow. I apologize for my lack of intelligence, but can you put it into a more simple language? n-dimensional means to me, with n=5, something like:

var = (5 times, 6 chunks, 10 layers, 1000 columns, 50 rows), or
var = fltarr(5, 6, 10, 1000, 50)

and n must be <= 8 in IDL.

Or, do you mean FLTARR(n, n_points), which is a 2D array? If so, this will be much simplier.
But based on my very limited understanding of your problem, I can propose that you use SIZE to get the dimensions of the original array, then create "discrete boxes", for which you could probably use REBIN (JD? are you here?). WHERE can tell you where a given point is.

I would not post this reply but I see nobody else replying, so decided to indicate to Peter that he's beed heard. Where are all the smart folks this morning?

Cheers,
Pavel

Peter Thorne wrote:
>
> Apologies if this is trivial, but I have been working for about 8 hours
> on it and can't even begin to see how to code it in IDL. Any help very
> gratefully received. I am setting up a function which receives the
> location of points in n-dimensional space for m fields, as well as their
> weights. On call the function does not know the size of any of these
> dimensions.
>
> Simplifying to m=1 (this m dimension should be trivial) the input is:
>
> an array of size (n x npoints) locations of each point in the
> n-dimensional space
>
> a vector of size (npoints) the weights.
>
> Now, what I want to be able to do is re-bin these points into an
> n-dimensional discretized space array which encloses all points. I can
> work out the limits of this space by simply finding min and max in each
> of the n dimensions of the location array. I then need to split this
> grid into nbox n-dimensional discrete boxes (say 50 divisions per
> dimension, boxes need not be of equal size in each dimension) and place

> the respective points in their boxes in this finite representation,
> weighted by weight (trivial).
>
> At present I am having two major problems:
>
> 1. How to declare this discretized array and the limits in n-dimensional
> space of each box into which I bin my values given that I know n and
> nbox.
>
> and doubtless much more difficult to overcome:
>
> 2. How to sensibly code selection criteria to ascertain whether a
> particular value belongs to a particular grid box.
>
> Perhaps it is not possible to code without an a priori knowledge of n -
> the dimensionality of the problem to know how many for loops to use? I
> can't see how where statements could be used.
>
> Any help or useful pointers very gratefully received.
>
> Thanks
>
> Peter

Subject: Re: Help setting up an array
Posted by Paul van Delst on Wed, 28 Mar 2001 17:51:56 GMT
View Forum Message <> Reply to Message

As Pavel said, I assume you mean FLTARR(2, 53, 6, 4, 6) when you talk of 5-dimensional space.

To create n-dimensional arrays with n unknown until run time, use MAKE_ARRAY.

To bin up all the values, it sounds like a coordinate conversion might be the easiest? If I understand correctly, and using a 3-D example, what you want to do is bin up values so that all points in a 3-D box (defined by the size of the division per dimension) would be lumped into a (weighted) single element of the re-binned space? Sorta like a multidimensional HISTOGRAM? The coordinate conversion would use the actual array indices along with the division value you pick for each dimension. Maybe you could even use HISTOGRAM? The help doesn't put a limit on the dimensions allowed in the input.... other out there would know better. The HISTOGRAM King, JD, might have an opinion.

If the above is a total load of bollocks compared to what you wnat to do then I will say I found your explanation of the problem difficult to follow - a sure sign you've been staring at it for too long. :o) A simpler example for us regular old 3-d creatures may provide more insight. For yourself as well as others.

paulv

Peter Thorne wrote:
>
> Apologies if this is trivial, but I have been working for about 8 hours
> on it and can't even begin to see how to code it in IDL. Any help very
> gratefully received. I am setting up a function which receives the
> location of points in n-dimensional space for m fields, as well as their
> weights. On call the function does not know the size of any of these
> dimensions.
>
> Simplifying to m=1 (this m dimension should be trivial) the input is:
>
> an array of size (n x npoints) locations of each point in the
> n-dimensional space
>
> a vector of size (npoints) the weights.
>
> Now, what I want to be able to do is re-bin these points into an
> n-dimensional discretized space array which encloses all points. I can
> work out the limits of this space by simply finding min and max in each
> of the n dimensions of the location array. I then need to split this
> grid into nbox n-dimensional discrete boxes (say 50 divisions per
> dimension, boxes need not be of equal size in each dimension) and place
> the respective points in their boxes in this finite representation,
> weighted by weight (trivial).
>
> At present I am having two major problems:
>
> 1. How to declare this discretized array and the limits in n-dimensional
> space of each box into which I bin my values given that I know n and
> nbox.
>
> and doubtless much more difficult to overcome:
>
> 2. How to sensibly code selection criteria to ascertain whether a
> particular value belongs to a particular grid box.
>
> Perhaps it is not possible to code without an a priori knowledge of n -
> the dimensionality of the problem to know how many for loops to use? I
> can't see how where statements could be used.
>
> Any help or useful pointers very gratefully received.
>
> Thanks
>
> Peter

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
paul.vandelst@noaa.gov Alexander Pope.

Subject: Re: Help setting up an array
Posted by Mark Fardal on Wed, 28 Mar 2001 18:11:28 GMT
View Forum Message <> Reply to Message

> Now, what I want to be able to do is re-bin these points into an
> n-dimensional discretized space array which encloses all points. I can

You mean something like this? I'm sure there are many ways to do it.

```
pro thorne, x, w, box
;x[ndim, npts]
;w[npts]

npts = n_elements(x[0,*])
ndim = n_elements(x[*,0])

xmin = fltarr(ndim)
xmax = fltarr(ndim)
dx = fltarr(ndim)
for i = 0, ndim-1 do begin
  xmin[i] = min(x[i,*])
  xmax[i] = max(x[i,*])
endfor
dx[*] = 1.  ;how is this determined?
ngrid = ceil( (xmax - xmin) / dx + 1.e-4 )

ibox = lonarr(npts)
nmult = 1
for i = 0, ndim-1 do begin
  ibox = ibox + nmult * fix( ( x[i,*] - xmin[i] ) / dx[i] )
  nmult = nmult * ngrid[i]
endfor

;this could be improved with histogram?
box = fltarr(nmult)
for j = 0, npts-1 do begin
  box[ibox[j]] = box[ibox[j]] + w[j]
endfor
```

```
string = 'box = reform(box, '
for i = 0, ndim-2 do string = string + string(ngrid[i], ', ', format='(i0,a)')
string = string + string(ngrid[i], ')', format='(i0,a)')
checkthisval = execute(string)

end
```

Subject: Re: Help setting up an array
Posted by Peter Thorne on Wed, 28 Mar 2001 21:54:01 GMT
View Forum Message <> Reply to Message

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

Subject: Re: Help setting up an array
Posted by John-David T. Smith on Wed, 28 Mar 2001 22:16:24 GMT
View Forum Message <> Reply to Message

Peter Thorne wrote:
>
> Apologies if this is trivial, but I have been working for about 8 hours
> on it and can't even begin to see how to code it in IDL. Any help very
> gratefully received. I am setting up a function which receives the
> location of points in n-dimensional space for m fields, as well as their
> weights. On call the function does not know the size of any of these
> dimensions.
>
> Simplifying to m=1 (this m dimension should be trivial) the input is:
>
> an array of size (n x npoints) locations of each point in the
> n-dimensional space
>
> a vector of size (npoints) the weights.
>
> Now, what I want to be able to do is re-bin these points into an
> n-dimensional discretized space array which encloses all points. I can
> work out the limits of this space by simply finding min and max in each
> of the n dimensions of the location array. I then need to split this
> grid into nbox n-dimensional discrete boxes (say 50 divisions per
> dimension, boxes need not be of equal size in each dimension) and place
> the respective points in their boxes in this finite representation,
> weighted by weight (trivial).
>
> At present I am having two major problems:
>
> 1. How to declare this discretized array and the limits in n-dimensional
> space of each box into which I bin my values given that I know n and
> nbox.
>
> and doubtless much more difficult to overcome:
>
> 2. How to sensibly code selection criteria to ascertain whether a
> particular value belongs to a particular grid box.
>
> Perhaps it is not possible to code without an a priori knowledge of n -
> the dimensionality of the problem to know how many for loops to use? I
> can't see how where statements could be used.

It seems what you are trying to do is take a list of coordinates and values and construct a data cube or hyper-cube. For a concrete example, consider a list of tuples of the form:

x,y,z,density

You'd like to create a 3-d array with enough elements to adequately represent the density cube. This immediately begs the question: how is the density sampled, regularly or irregularly? If the answer is regularly, the solution is trivial:

```
s=size(c,/DIMENSIONS)
n=intarr(s[0])
for i=s[0]-1,0,-1 do begin
  ind=(c[i,*]-min(c[i,*]))/spacing[i]
  n[i]=max(ind)+1
  inds=n_elements(inds) eq 0?ind:ind+n[i]*inds
endfor
array=make_array(DIMENSION=n,type=size(w,/TYPE))
array[inds]=w
```

where "c" is the 3xn array of x,y,z coordinates (or dxn for your dimensionality), "w" is the n point vector of densities or weights, etc. And "spacing" is the n point vector of the regular spacing in each dimension.

E.g. suppose x runs from 0 to 20 with spacing .5, y runs from 0 to 50 with spacing 1, and z runs from 10 20 with spacing 2.

You'd set spacing=[.5,1,2] and you'd get a 41x51x6 array, with all the values plugged into the right place. Note that all this is doing is rearranging data which could otherwise have been constructed into a data cube in the first place.

If, as I suspect, you'd like to resample irregularly gridded higher-dimensional data, you'll need to think about what sort of sampling, and onto what size mesh you'd like to sample. There is no fundamental answer to these questions within the dataset itself, though you could of course come up with quantitative heuristics.

JD

Subject: Re: Help setting up an array
Posted by Craig Markwardt on Wed, 28 Mar 2001 22:49:08 GMT
View Forum Message <> Reply to Message

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.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

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 169 times

Subject: Re: Help setting up an array
Posted by davidf on Thu, 29 Mar 2001 04:11:46 GMT
View Forum Message <> Reply to Message

JD Smith (jdsmith@astro.cornell.edu) writes:

> 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:

Oh, my gosh. *Surely* something is missing here. The documentation
header is 10 times longer than the code! You didn't put this

in a SAVE file, did you, and truncate the thing?

Cheers,

David

P.S. Let's just say that seals it. I'm retiring.
Who's even gonna look at MPI_PLOT. :-(

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Help setting up an array
Posted by Peter Thorne on Thu, 29 Mar 2001 08:47:39 GMT
View Forum Message <> Reply to Message

Wow!

Not sure what else one can say really. That would have taken me
literally weeks to come up with. It seems to be exactly what was
required. I might even complete my PhD on time now ...

Many thanks to all for useful and pertinent comments, and especially to
JD for this function.

Peter

Subject: Re: Help setting up an array
Posted by Martin Schultz on Thu, 29 Mar 2001 09:33:45 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> P.S. Let's just say that seals it. I'm retiring.
> Who's even gonna look at MPI_PLOT. :-(
>
Don't retire: write the widget interface for n-dimensional histograms
so that even mortals can use it ;-)

Cheers,

Martin

--
[[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[[

Subject: Re: Help setting up an array
Posted by Jaco van Gorkom on Thu, 29 Mar 2001 11:16:31 GMT
View Forum Message <> Reply to Message

Hi,
this question seemed interesting yesterday night, when no answers were present yet. Now, finally getting back to a network with IDL license and news reader, JD's little function takes all the fun out of it. So I am not even going to test the bits of code I came up with during dinner. Posting it anyway for the sake of completeness:

Peter Thorne wrote:
>
> Apologies if this is trivial, ...
Trivial?? Well, that is *one* way to put it, I suppose. It could be done cryptically in just two statements:

```
numberdensity = reform(histogram((pos-rebin(minpos,n,npoints,/sample))*$
  rebin(nbox^indgen(n)/(maxpos-minpos)/nbox,n,npoints,/sample) ,min=0,$

 max=nbox^n-1,binsize=1,reverse_indices=R),replicate(nbox,n), /overwrite)
weighteddensity = reform(convol(total(weight[R[R[0]:*]],/cumulative)$
  [R[0:R[0]-1]-R[0]],[1,-1],center=0),replicate(nbox,n),/overw rite)
```

Now the long version...
> I am setting up a function which receives the
> location of points in n-dimensional space for m fields, as well as their
> weights. On call the function does not know the size of any of these
> dimensions.
>
> Simplifying to m=1 (this m dimension should be trivial) the input is:
>
> an array of size (n x npoints) locations of each point in the
> n-dimensional space
let's call this array POS. And use the SIZE() function to find out N and

NPOINTS.
> a vector of size (npoints) the weights.
So this is the vector WEIGHT.
> Now, what I want to be able to do is re-bin these points into an
> n-dimensional discretized space array which encloses all points. I can
> work out the limits of this space by simply finding min and max in each
> of the n dimensions of the location array.
This gives two vectors, each of size (npoints): I name them MINPOS and MAXPOS.
> I then need to split this
> grid into nbox n-dimensional discrete boxes (say 50 divisions per
> dimension, boxes need not be of equal size in each dimension)
I will use the scalar NBOX as the number of divisions per dimension, e.g.
50. Although there is no real reason to take this number equal in each dimension, of course.
> and place
> the respective points in their boxes in this finite representation, ...

What you are trying to do here - counting the number of points in each of a
large set of equally spaced bins - is basically to take a histogram over the
positions. This is very fortunate, since the built-in function HISTOGRAM()
seems to be the most often (mis-)used routine for vectorising all kinds of
operations in IDL. It can do almost anything for you, without ever resorting
to FOR-loops. And what HISTOGRAM cannot accomplish by itself, can be done by
deciphering its keyword REVERSE_INDICES.

Here we don't need to do anything very fancy with HISTOGRAM. Basically, a
simple histogram is all we need, be it over N dimensions. HISTOGRAM itself
appears to be a 1D routine, but there is a technique for using it over
multiple dimensions, used in HIST_2D for the 2D case (part of the IDL
distribution). Now the best path (and highly recommended!) would be
to generalise HIST_2D to PT_HIST_ND for the N-dimensional case. Taking the
path of least resistance, let us just use their technique. Basically the
trick is to scale all the position coordinates such that they exactly fit
between 0 and NBOX (50), giving a box size which is normalised to 1. Then
combine the N position coordinates into one big scalar using some kind

of
"NBOXimal system", much like different numbers 0 to 9 can be combined in
the
decimal system to form two-, three-, or N-digit numbers by multiplying
the
Nth number by 10^(N-1). This big single scalar position coordinate is
actually
the same as the linear subscripts of the N-dimensional array of boxes,
and
thus also equal to the linear subscript of the N-dimensional histogram
that
we are after.

That part was not very clear, was it? Anyway, here goes:
First make all position coordinates start at zero
```
  pos = pos - rebin(minpos, n, npoints, /sample)
```
Normalise them to the boxsixe:
```
  boxsize = (maxpos - minpos) / nbox
  pos = pos / rebin(boxsize, n, npoints, /sample)
```
Finally, combine all the N coordinates into one super-coordinate
(NBOXimal):
```
  pos = pos * rebin(nbox^indgen(n), n, npoints, /sample)
  pos = total(pos, 2)   ; sum over the N dimensions
```
Now pos is a vector of size (npoints). Of course some of these
statements
could have been combined for efficiency.

Now counting the number of points in each box is easy:
```
  numberdensity = histogram(pos, min=0, max=nbox^n-1, binsize=1,
reverse_indices=R)
```

> weighted by weight (trivial).
>

Weighting by weight is now not as trivial as it sounds. However,
REVERSE_INDICES comes to the rescue! This keyword will contain the
information on which points contributed to which box, although the way
it is
coded into one array tends to give me at least a slight headache. After
initialising the destination array to the same number of elements as
NUMBERDENSITY, you'd want to do something like
```
  for i=0, n_elements(numberdensity)-1 then $
    if R[i] ne R[i+1] then $
      weighteddensity[i] = total(weight[R[R[i]:R[i+1]-1]])
```
This just adds up the contributing elements of the WEIGHT array for each
box.

Several loopless, conditionless versions of the weighting procedure are

possible and usually faster:
```
  weights_cumul = total(weight[R[R[0]:*]],/cumulative)
  weighteddensity = weights_cumul[R[1:R[0]-1]-R[0]] $
                  - weights_cumul[R[0:R[0]-2]-R[0]]
```

If space has less than nine dimensions, then the space dimensions of the array of boxes can be represented by IDL array dimensions:
```
  if n le 8 then $
    weighteddensity =
reform(weighteddensity,replicate(nbox,n),/overwrite)
```
Note that apart from this there is (almost) no limitation on N!

Just out of curiosity: did you create this little puzzle just to test
our
brain cells, or is there a real-world application for this problem?

cheers,
  Jaco

Subject: Re: Help setting up an array
Posted by Peter Thorne on Thu, 29 Mar 2001 14:58:57 GMT
View Forum Message <> Reply to Message

Jaco van Gorkom wrote:

> Just out of curiosity: did you create this little puzzle just to test
> our
> brain cells, or is there a real-world application for this problem?

It is a real world problem, believe it or not! We have a number of observational parameters which we wish to regress against possible modelled causes in a system containing "noise". Output from this regression is effectively a cloud of potential solution points, an n-dimensional PDF (probability density function) ellipsoid. Previously we have only considered ellipsoids as single pieces of data. However, in this system we wish to assess the consistency of the model system and therefore need to intercompare m n-dimensional ellipsoids (where m distinct realisations are made through the regression analysis). Effectively we need to set up a system whereby the null hypothesis is that all m fields gained are equivalent (are sub-sampled from some true population). To gain a quantitative measure of this statistic it is required to integrate the fields over the n-dimensional phase space which is common to the m fields and gain the maximum probability function from the m fields for this integral. Maximum because the ellipsoids are not expected to have equal variance, distributions or orientation in the regression phase space.

Well, you did ask ...

Subject: Re: Help setting up an array
Posted by Jaco van Gorkom on Thu, 29 Mar 2001 15:33:49 GMT
View Forum Message <> Reply to Message

Peter Thorne wrote:
>
> Jaco van Gorkom wrote:
>
>> Just out of curiosity: did you create this little puzzle just to test
>> our
>> brain cells, or is there a real-world application for this problem?
>
> It is a real world problem, believe it or not! We have a number of
> observational parameters which we wish to regress against possible
> modelled causes in a system containing "noise". Output from this
> regression is effectively a cloud of potential solution points, an
> n-dimensional PDF (probability density function) ellipsoid. Previously
> we have only considered ellipsoids as single pieces of data. However, in
> this system we wish to assess the consistency of the model system and
> therefore need to intercompare m n-dimensional ellipsoids (where m
> distinct realisations are made through the regression analysis).
> Effectively we need to set up a system whereby the null hypothesis is
> that all m fields gained are equivalent (are sub-sampled from some true
> population). To gain a quantitative measure of this statistic it is
> required to integrate the fields over the n-dimensional phase space
> which is common to the m fields and gain the maximum probability
> function from the m fields for this integral. Maximum because the
> ellipsoids are not expected to have equal variance, distributions or
> orientation in the regression phase space.
>
> Well, you did ask ...

Aha, I see.
Let me just compliment you on the excellent description of the
essentials
which you gave in the original post.
Now, as for your notion of "real world"..... :-)

Subject: Re: Help setting up an array
Posted by davidf on Thu, 29 Mar 2001 15:34:11 GMT
View Forum Message <> Reply to Message

Peter Thorne (peter.thorne@uea.ac.uk) writes:
>
> It is a real world problem, believe it or not! We have a number of
> observational parameters which we wish to regress against possible
> modelled causes in a system containing "noise". Output from this
> regression is effectively a cloud of potential solution points, an
> n-dimensional PDF (probability density function) ellipsoid. Previously
> we have only considered ellipsoids as single pieces of data. However, in
> this system we wish to assess the consistency of the model system and
> therefore need to intercompare m n-dimensional ellipsoids (where m
> distinct realisations are made through the regression analysis).
> Effectively we need to set up a system whereby the null hypothesis is
> that all m fields gained are equivalent (are sub-sampled from some true
> population). To gain a quantitative measure of this statistic it is
> required to integrate the fields over the n-dimensional phase space
> which is common to the m fields and gain the maximum probability
> function from the m fields for this integral. Maximum because the
> ellipsoids are not expected to have equal variance, distributions or
> orientation in the regression phase space.
>
> Well, you did ask ...

Job Wanted:

Former IDL programmer seeking position with light programming
responsibilities. Has modest ability to get colors right.
Tennis court nearby a distinct advantage. All inquires kept
confidential.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Help setting up an array
Posted by John-David T. Smith on Thu, 29 Mar 2001 16:47:00 GMT
View Forum Message <> Reply to Message

Peter Thorne wrote:
>
> Wow!

>
> Not sure what else one can say really. That would have taken me
> literally weeks to come up with. It seems to be exactly what was
> required. I might even complete my PhD on time now ...
>

Perhaps you can write one of *my* chapters for me then. I also clearly need some 5-dimensional dataset to include... "and the 5-dimensional joint density function is projected onto a fractal hyper-cube with d=2.25..." Something along those lines would impress, I'm sure.

JD

Subject: Re: Help setting up an array
Posted by Pavel A. Romashkin on Thu, 29 Mar 2001 17:25:47 GMT
View Forum Message <> Reply to Message

Hey JD, I always knew you'll be the one who would solve this, anyway (see my very first posting above).
David, you don't have to worry. There is only one JD here, and there are far too many questions being asked for him to answer alone.
BTW, JD, why don't you give us here a call once you are done with your project? :-)

Cheers,
Pavel

JD Smith wrote:
>
> Peter Thorne wrote:
>>
>> Wow!
>>
>> Not sure what else one can say really. That would have taken me
>> literally weeks to come up with. It seems to be exactly what was
>> required. I might even complete my PhD on time now ...
>>
>
> Perhaps you can write one of *my* chapters for me then. I also clearly
> need some 5-dimensional dataset to include... "and the 5-dimensional
> joint density function is projected onto a fractal hyper-cube with
> d=2.25..." Something along those lines would impress, I'm sure.
>
> JDD

Subject: Re: Help setting up an array
Posted by Pavel A. Romashkin on Thu, 29 Mar 2001 17:31:24 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Job Wanted:
>
> Former IDL programmer seeking position with light programming
> responsibilities. Has modest ability to get colors right.
> Tennis court nearby a distinct advantage. All inquires kept
> confidential.

Confidential

Once you get that position, David, do you think you can use someone to dust your desk, brew coffee and make sure IDL is loaded by the time you start in the morning? Can also invoke Heap_gc when told so to take this blame off of you. Minimum wage is ok.

Cheers,
Pavel

Subject: Re: Help setting up an array
Posted by Craig Markwardt on Thu, 29 Mar 2001 21:23:10 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:
>
> 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:

Doh! You beat me to it. I still have a bunch of tricks up my sleeves for a really nice advanced histogramming function, so I'll still plug away when I have time...

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response

---------- ----------

Subject: Re: Help setting up an array
Posted by davidf on Thu, 29 Mar 2001 21:56:27 GMT
View Forum Message <> Reply to Message

Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:

> Doh! You beat me to it. I still have a bunch of tricks up my sleeves
> for a really nice advanced histogramming function, so I'll still plug
> away when I have time...

Yeah, I'm working on something, too. I just keep running
out of fingers as I try to count up the reverse indices. :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Help setting up an array
Posted by Pavel A. Romashkin on Fri, 30 Mar 2001 16:32:20 GMT
View Forum Message <> Reply to Message

You folks just won't give up. Oh well. I admit to my inferiority in this
case, and when I need a Histogram, I will just email the H-King, and
beers are on me when JD and I finally meet :-)

Cheers,
Pavel

Craig Markwardt wrote:
>
> JD Smith <jdsmith@astro.cornell.edu> writes:
>>
>> 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:
>
> Doh! You beat me to it. I still have a bunch of tricks up my sleeves
> for a really nice advanced histogramming function, so I'll still plug
> away when I have time...
>
> Craig
