Subject: Re: I need some histogram magic - gridding very large dataset
Posted by rjp23 on Thu, 02 May 2013 09:45:52 GMT
View Forum Message <> Reply to Message

Ok, after re-reading the tutorial I'm getting closer.

```
histogram = HIST_ND(TRANSPOSE([[lon],[lat]]), $
        [1,1], MIN=[-180, -90], $
        MAX=[180, 90],REVERSE_INDICES=ri)

grid_mean=fltarr(361, 181)
grid_mean[*]=!values.f_nan

FOR i=0L,N_ELEMENTS(histogram)-1 DO IF(histogram[i] GT 0) THEN $
        grid_mean[i] = mean(data[ri[ri[i]:(ri[i+1]-1)]])
```

The only thing that's let me a bit confused is why does hist_nd return a (361, 181) array? Isn't that 1 element too large in both direction? If the min and max is -180 and 180, that's 361 values but the bins range between those values e.g. -180 to -179, -179 to -178..., 179 to 180 so there should only be 360 bins (which is what I want).

Or is min/max defined differently? Is max the lower bound of the final bin, so the actual maximum value is max+binsize?

Subject: Re: I need some histogram magic - gridding very large dataset
Posted by John Correira on Thu, 02 May 2013 13:44:00 GMT
View Forum Message <> Reply to Message

On 05/02/2013 05:45 AM, rjp23@le.ac.uk wrote:
>
> Or is min/max defined differently? Is max the lower bound of the
> final bin, so the actual maximum value is max+binsize?
>

I've run into this issue myself. I think the behavior of HISTOGRAM and the name of the MAX keyword is somewhat counter intuitive here. See if this thread helps:

https://groups.google.com/d/msg/comp.lang.idl-pvwave/RuGGGF_iufc/d4OJg9ZOrZcJ

Regards,

John

Subject: Re: I need some histogram magic - gridding very large dataset
Posted by Craig Markwardt on Thu, 02 May 2013 14:10:01 GMT
View Forum Message <> Reply to Message

On Thursday, May 2, 2013 5:45:52 AM UTC-4, rj...@le.ac.uk wrote:
> Ok, after re-reading the tutorial I'm getting closer.
...
>
> The only thing that's let me a bit confused is why does hist_nd return a (361, 181) array? Isn't that 1 element too large in both direction? If the min and max is -180 and 180, that's 361 values but the bins range between those values e.g. -180 to -179, -179 to -178..., 179 to 180 so there should only be 360 bins (which is what I want).
>
>
>
> Or is min/max defined differently? Is max the lower bound of the final bin, so the actual maximum value is max+binsize?

I've run into similar problems. Perhaps the attached GEOSPLAT() function will do what you want. It does some trickery to avoid that extra bin at the end.

It asks for a LON and LAT array, and also an EXPOsure array which can be any additive weighting for your points. In my field it's the amount of exposure time in seconds, but it can be anything, or just an array of 1's in order to compute a straight average. It also returns the total exposure array and a sample variance array.

Craig

```
;
; GEOSPLAT - average LAT/LON quantities with weighting
;
;  LON - longitude value (range 0,360)
;  LAT - latitude value (range -90,90)
;  VAL - input values to be averaged
;  EXPO - input exposure weighting for each VAL
;  XRANGE - LON range of interest (example [0,360])
;  YRANGE - LAT range of interest (example [-90,90])
;  XBINSIZE - LON bin size [deg]
;  YBINSIZE - LAT bin size [deg]
;  MINSAMP - minimum number of samples per bin to do averaging
;  EXPOSURE - output array giving exposure for each bin
;  VARIANCE - output array giving sample standard deviation for each bin
;
;  RETURNS: array average for each bin
;
function geosplat, lon, lat, val, expo, xrange=xrange0, yrange=yrange0, $
                xbinsize=xbinsize, ybinsize=ybinsize, minsamp=minsamp, $
```

```
                variance=hh2, exposure=ee

forward_function arg_present, hist_2dr

if n_elements(xrange0) LT 2 then xrange0 = [0,359.9999d]
if n_elements(yrange0) LT 2 then yrange0 = [-90,89.9999d]
xrange = xrange0(0:1)
yrange = yrange0(0:1)
if n_elements(xbinsize) LT 1 then xbinsize = 1
if n_elements(ybinsize) LT 1 then ybinsize = 1
if n_elements(minsamp) LT 1 then minsamp = 1
if n_elements(expo) LT n_elements(val) then expo = val*0 + 1.
if arg_present(hh2) then dohh2 = 1 else dohh2 = 0

;; Some skull-jiggery to get the bin sizing right, so that we
;; don't get an extra orphan bin at the edge.
nxbins0 = (xrange(1) - xrange(0)) / xbinsize(0)
nybins0 = (yrange(1) - yrange(0)) / ybinsize(0)
dx = nxbins0 - floor(nxbins0)
dy = nybins0 - floor(nybins0)
if dx GT 0 AND dx LT 0.001d then xbinsize = (xrange(1)-xrange(0))/round(nxbins0)
if dy GT 0 AND dy LT 0.001d then ybinsize = (yrange(1)-yrange(0))/round(nybins0)

hh = hist_2dr(lon, lat, reverse=rr0, $
              min1=xrange(0), max1=xrange(1), bin1=xbinsize(0), $
              min2=yrange(0), max2=yrange(1), bin2=ybinsize(0))
hh = hh * (val(0)*0)
ee = hh
if dohh2 then hh2 = hh
for i = 0, n_elements(hh)-1 do begin
    if rr0(i+1) - rr0(i) GT minsamp(0) then begin
        ex = expo(rr0(rr0(i):rr0(i+1)-1))
        vv = val(rr0(rr0(i):rr0(i+1)-1))
        hh(i) = total(vv*ex)
        ee(i) = total(ex)
        if dohh2 then hh2(i) = total(vv^2*ex)
    endif
endfor

hh = hh / (ee>1e-20)
if dohh2 then hh2 = hh2 / (ee>1e-20)
if dohh2 then hh2 = hh2-hh^2

;; Unexposed portions are NAN
wh = where(ee LT 1d-20, ct)
if ct GT 0 then begin
   hh(wh) = !values.d_nan
   if dohh2 then hh2(wh) = !values.d_nan
```

```
  endif

  return, hh
end
```

Subject: Re: I need some histogram magic - gridding very large dataset
Posted by John Correira on Thu, 02 May 2013 14:42:57 GMT
View Forum Message <> Reply to Message

On 05/02/2013 10:10 AM, Craig Markwardt wrote:
> I've run into similar problems. Perhaps the attached GEOSPLAT()
> function will do what you want. It does some trickery to avoid that
> extra bin at the end. It asks for a LON and LAT array, and also an
> EXPOsure array which can be any additive weighting for your points. In
> my field it's the amount of exposure time in seconds, but it can be
> anything, or just an array of 1's in order to compute a straight
> average. It also returns the total exposure array and a sample
> variance array. Craig

Below was my attempt at solving this problem. It allows for an arbitrary function to be applied to each bin. It's a work in progress. I haven't looked at it in awhile so I can't remember how well it worked the last time I tried to use it :-). Suggested improvements welcome.

Regards,

John

%%%%%%%%%%%%%%%%

```
FUNCTION cpi_2d_histobin, x, y, value, funct, $
  funct_arguments=funct_arguments, $
  BINSIZE=binsize, $
  MIN=mn, $
  MAX=mx, $
  NBINS=nbins, $
  emptybinvalue=emptybinvalue, $
  hdata=hdata, $
  x_size = x_size, $
  y_size = y_size, $
  debug=debug

  COMPILE_OPT IDL2, STRICTARRSUBS

  if ~KEYWORD_SET(debug) THEN BEGIN
    CATCH, theError
    IF theError NE 0 THEN BEGIN
```

```
      CATCH, /CANCEL
      print, '% '+!ERR_STRING
      return, -1
    ENDIF
  ENDIF

  if N_ELEMENTS(emptybinvalue) eq 0 then begin
    tname = size(value, /TNAME)
    if tname eq 'FLOAT' then emptybinvalue = !values.f_nan
    if tname eq 'DOUBLE' then emptybinvalue = !values.d_nan
  endif

  data = transpose([[x],[y]])

  imx = max(data,DIMENSION=2,MIN=imn)

  if n_elements(mx) eq 0 then mx=imx
  if n_elements(mn) eq 0 then mn=imn

  if ((N_ELEMENTS(binsize) GT 0) AND (N_ELEMENTS(nbins) GT 0)) THEN $
    print, ' BINSIZE and NBINS keywords conflicting. Ignoring NBINS.'

;  if (N_ELEMENTS(binsize) eq 0) then binsize = (mx-mn)/nbins
  h =
 hist_nd(data,binsize*(1+(MACHAR()).eps),MIN=mn,MAX=mx+binsiz e,REVERSE_INDICES=ri)


  nx = (size(h))[1]
  ny = (size(h))[2]

  x_size = (mx-mn+1)[0]/float(nx)
  y_size = (mx-mn+1)[1]/float(ny)

  val = h
  val *= emptybinvalue

  if N_ELEMENTS(funct_arguments) gt 0 then begin
    for i=0L, nx-1 do begin
      for j=0L, ny-1 do begin
        if h[i,j] gt 0 then begin
          ind = i+nx*j
          val[i,j] = CALL_FUNCTION(funct,value[ri[ri[ind]:ri[ind+1]-1]], $
            _STRICT_EXTRA=funct_arguments)
        endif else continue
      endfor
    endfor
  endif else if N_ELEMENTS(funct_arguments) eq 0 then begin
     for i=0L, nx-1 do begin
```

```
      for j=0L, ny-1 do begin
        if h[i,j] gt 0 then begin
          ind = i+nx*j
          val[i,j] = CALL_FUNCTION(funct,value[ri[ri[ind]:ri[ind+1]-1]])
          endif else continue
      endfor
    endfor
  endif

  val =  val[0:n_elements(val[*,0])-2,indgen(N_ELEMENTS(val[0,*])-1)]

  if arg_present(hdata) then hdata = h

  return, val

END
```