Subject: Resampling data with irregular time base
Posted by krieger on Sat, 05 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

I have data with an irregular time base, which I would like to resample
in a regular spaced time base. How can I average over all original data
points in each interval of the new time vector without resorting to a
FOR loop?
Currently I am using this horrible kludge:

```
deltat = newtime[1] - newtime[0]
FOR n=0, n_elements(newtime)-1 DO BEGIN
  index = where((oldtime GT (newtime[n]-deltat/2.)) AND $
                (oldtime LE (newtime[n]+deltat/2.)), $
                count)
  IF count GT 0 THEN newdata[n] = total(olddata[index]) / count
ENDFOR
```

Any idea how to transform this in vectorized IDL code? At the moment I
see no way apart from writing the function in C and calling it by
linkimage.

Best

Karl

--
To reply by email please replace domain .NOSPAM by .de in reply address

Subject: Re: Resampling data with irregular time base
Posted by Paul Krummel on Wed, 09 Jun 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <7jid9q$4td$1@news.lth.se>,
  Struan Gray <struan.gray@sljus.lu.se> wrote:
>
> Use the HISTOGRAM function with the REVERSE_INDICES keyword on
your
> array of time values. You can use the MAX, MIN and BINSIZE keywords
> to define the start, stop and interval times of the new timebase.
> Then, for each of those intervals the array returned by
> REVERSE_INDICES will tell you the elements of your original data which
> lie in that interval, so it's easy to add them up. You can extract the
> normalisation divisor from the number of elements pointed to by the
> reverse indices array, or from the value of the relevant bin of the
> histogram itself.
>

> Struan
>
The routine below may be of some use to people in this group. I wrote it as a general averaging routine, and like Straun suggested, it uses the reverse_indices keyword to histogram. The code was specifically written for averaging data measured by research aircraft (Temp, moisture, liquid water etc) against the corresponding pressure. I wrote it some years ago and perhaps it could be improved?
Anyway, let me know any problems, improvements etc.
Cheers Paul

```
;+
; NAME:
; INDEP_AVERAGE
;
; PURPOSE:
; This procedure will average data that has been "binned"
; according to an independent data variable. An example
; application is with aircraft data, where the aircraft
; is flying up and down through the atmosphere. This
; routine would be used to find an average temperature
; profile through the atmosphere where all the temperature
; measurements falling into a certain pressure range
; (or bin range) are averaged.
;
;
; CATEGORY:
; Statistics.
;
; CALLING SEQUENCE:
; INDEP_AVERAGE, Indep, Data, Bin, Max_val, Min_val, $
;          Indep_mid, Data_mid, Count_mid, $
;          Count_non_zero, $
;          N_mid_levels, VARIANCE=variance
;
; INPUTS:
; Indep:  an array containing the independent
;          variable ie pressure level.
; Data:   an array containing the data that is to
;          be averaged. This array must be the same
;          size as Indep!!
; Bin:    a scalar containing the bin size to be
;          averaged over. Same units as Indep.
; Max_val:    a scalar containing the maximum value
;             of the Indep variable. (NOTE: this has
;             to be a multiple of the 'bin' size)
; Min_val:    a scalar containing the minimum value
```

```
;             of the Indep variable. (NOTE: this has
;             to be a multiple of the 'bin' size)
;
; KEYWORD PARAMETERS:
; VARIANCE: Set this keyword to return an array
;             containing the variance for each
;             averaged level. Has size N_mid_levels.
;
; OUTPUTS:
; This procedure will return the following:
; Indep_mid:  This is an array containing the center
;             value of each of the bins. Has size
;             N_mid_levels. Float.
; Data_mid:   This is an array containing the averaged
;             data. Has size N_mid_levels. Float.
; Count_mid:  This is an array containing the number
;             of points used in the average for each
;             averaged level. Has size N_mid_levels. Float.
; Count_non_zero: This is an array containing the number of
;             points that are non-zero at each averaged
;             level. Has size N_mid_levels. Float.
; N_mid_levels:   This is the number of averaged levels
;             (NOTE: this count starts at zero). Integer.
;
; PROCEDURE:
; Uses reverse indices option of the histogram keyword
; to find the data for each bin range.
;
; EXAMPLE:
; To find the average temperature profile in 10 hPa
; bins from aircraft data, where pressure and
; temperature are arrays:
; IDL> indep_average, pressure, temperature, 10, 1020, 700, $
;          press_mid, temp_av, count_mid, count_non_zero, $
;          n_mid_levels, VARIANCE=variance
; IDL> plot, temp_av, press_mid, yrange=[1020, 700]
;
; MODIFICATION HISTORY:
;  Written by: Paul Krummel, CSIRO Division of Atmospheric
;  Research, 17 September 1995.
; Variance keyword added on 4 March 1996 by PBK.
;-
;
PRO INDEP_AVERAGE, Indep, Data, Bin, Max_val, Min_val, $
       Indep_mid, Data_mid, Count_mid, Count_non_zero, $
       N_mid_levels, VARIANCE=variance, HELP=help
;
; =====>> HELP
```

```
;
on_error,2
if (N_PARAMS(0) LT 5) or keyword_set(help) then begin
   doc_library,'INDEP_AVERAGE'
   if N_PARAMS(0) LT 5 and not keyword_set(help) then $
     message,'Need at least 5 parameters, see above for usage.'
   return
endif
;
;
; ++++
; Find the counts for each independent bin range here.
; NOTE the '-bin/10' on the max_val, this is to stop
; the array count_mid being 1 larger than all the other
; arrays. This was happening because say min_val was 0,
; max_val 20 and bin 10, then histogram would look at
; values 0->9, 10->19 AND 20!!!!
count_mid=histogram(indep, binsize=bin, reverse_indices=r, $
                    min=min_val, max=max_val-bin/10)
;
; ++++
; Find the number of levels
n_mid_levels=(max_val-min_val)/bin - 1
print,'Number of independent levels is ',n_mid_levels
;
; ++++
; Calculate the middle independent value for each bin.
indep_mid=fltarr(n_mid_levels+1)
indep_mid[0]=min_val+bin/2.
for i=1,n_mid_levels do indep_mid[i]=indep_mid[i-1]+bin
;
; ++++
; Find the average of of the data for each bin here.
data_mid=fltarr(n_mid_levels+1)
variance=fltarr(n_mid_levels+1)
;
for i=0,n_mid_levels do begin
;   if r(i) ne r(i+1) then data_mid(i)= $
;   total(data(r(r(i):r(i+1)-1)))/count_mid(i)
; Make sure have non-zero data else get an error in moment!!
   count_nz=0
   if r[i] ne r[i+1] then $
      test_nz=where(data[r[r[i]:r[i+1]-1]],count_nz)
   if count_nz gt 0 then begin
; Make sure have more than 1 data point and that
; the data points are different else get an error
; in moment!!
      if (count_mid[i] gt 1) and (min(data[r[r[i]:r[i+1]-1]]) lt   $
```

```
                        max(data[r[r[i]:r[i+1]-1]])) then begin
        if r[i] ne r[i+1] then stats=moment(data[r[r[i]:r[i+1]-1]])
        if r[i] ne r[i+1] then data_mid[i]=stats[0]
        if (n_elements(VARIANCE) gt 0) then $
         if (r[i] ne r[i+1]) then variance[i]=stats[1]
      endif else begin
; If there is one data point this will set it to that
; value and divide by 1.
        if r[i] ne r[i+1] then $
         data_mid[i]=total(data[r[r[i]:r[i+1]-1]])/count_mid[i]
      endelse
   endif
endfor
;
; ++++
; Find the number of non-zero elements in each level.
count_non_zero=fltarr(n_mid_levels+1)
for i=0,n_mid_levels do begin
   cnt=0
   if r[i] ne r[i+1] then blah=where(data[r[r[i]:r[i+1]-1]] gt 0.0, cnt)
   count_non_zero[i]=cnt
endfor
;
; ++++
end
```