Subject: Subscripting help
Posted by khyde on Thu, 29 Oct 2015 14:44:23 GMT
View Forum Message <> Reply to Message

Hello,

I need to convert a large 2D array to a smaller 1D array by taking the mean of several pixels in the 2D array before putting them into known locations in the 1D array (or bin). The challenge is that the number of pixels per bin varies so I don't know how to do it without creating a large loop. I was hoping there was a way I could do it with subscripts, but I haven't been able to figure it out yet.

Here is an example of what I want to do using a loop. It works fine for this example, but my real 2D array is 8640x4320 and I need to move it to a ~5.6 million 1D array and it would take days to run it in a loop.

```
  ARR_2D = FINDGEN(9,10)        ; Input 2D array
  ARR_1D = FLTARR(18)            ; Output 1D array (bins)

; MAKE UP SUBSCRIPTS FOR THE ARRAY TO FIT INTO THE BINS (KNOWN BIN
LOCATIONS)
  SUBS = LONG(ARR_2D)
  S = [REVERSE(INDGEN(9)+1),INDGEN(9)+1]
  FOR N=0, N_ELEMENTS(S)-1 DO BEGIN
    IF N EQ 0 THEN FSUB = 0 ELSE FSUB = LSUB
    IF N EQ 0 THEN LSUB = S(N)-1 ELSE LSUB = FSUB + S(N)
    SUBS(FSUB:LSUB) = N
  ENDFOR


  ; LOOP METHOD
  FOR N=0, N_ELEMENTS(ARR_1D)-1 DO BEGIN
    OK = WHERE(SUBS EQ N, COUNT)
    IF COUNT GE 1 THEN ARR_1D(N) = MEAN(ARR_2D(OK))
  ENDFOR

 PRINT, ARR_2D
 PRINT
 PRINT, ARR_1D
```

Is there a faster way to get the same results? I was trying to figure out a way to do it with subscripts, but haven't had much success.

Thanks for your input.
Kim

Subject: Re: Subscripting help
Posted by greg.addr on Thu, 29 Oct 2015 16:23:21 GMT
View Forum Message <> Reply to Message

From the code you show, there's no need to make two loops: your first finds the indices - just apply them straight away to get the means, and forget SUBS.

It's possible there there's a way to do it without loops at all, but I can't figure out what you're trying to do. What is 'S' in the general case?

greg

Subject: Re: Subscripting help
Posted by Dick Jackson on Thu, 29 Oct 2015 16:23:50 GMT
View Forum Message <> Reply to Message

On Thursday, 29 October 2015 07:44:28 UTC-7, KH wrote:
> Hello,
>
> I need to convert a large 2D array to a smaller 1D array by taking the mean of several pixels in the 2D array before putting them into known locations in the 1D array (or bin). The challenge is that the number of pixels per bin varies so I don't know how to do it without creating a large loop. I was hoping there was a way I could do it with subscripts, but I haven't been able to figure it out yet.
>
> Here is an example of what I want to do using a loop. It works fine for this example, but my real 2D array is 8640x4320 and I need to move it to a ~5.6 million 1D array and it would take days to run it in a loop.
>
> [...]
>
> Is there a faster way to get the same results? I was trying to figure out a way to do it with subscripts, but haven't had much success.
>
> Thanks for your input.
> Kim

Hi Kim,

Your code might take days, maybe not, but there are faster ways. First thing I can think of is that Histogram() is your friend. This article is amazing, and worth a read:

http://www.idlcoyote.com/tips/histogram_tutorial.html

... but what you need here is Histogram() with REVERSE_INDICES, which is described concisely here:

http://www.exelisvis.com/docs/HISTOGRAM.html

Below is an amended version of your code that prints out a few more things than before, and gives two versions of the histogram method.

The creation of the SUBS array is a curious business, and if my guess is right, it may not be getting loaded with exactly what you want it to have (if S is meant to be how many values to average for each result in ARR_1D, then SUBS is not ending up exactly right).

```
S
  9  8  7  6  5  4  3  2  1  1  2  3  4  5  6  7  8  9

SUBS
  0  0  0  0  0  0  0  0  1
  1  1  1  1  1  1  1  2  2
  2  2  2  2  2  3  3  3  3
  3  3  4  4  4  4  4  5  5
  5  5  6  6  6  7  7  8  9
 10 10 11 11 11 12 12 12 12
 13 13 13 13 13 14 14 14 14
 14 14 15 15 15 15 15 15 15
 16 16 16 16 16 16 16 16 17
 17 17 17 17 17 17 17 17 17
```

If you can clarify that, there may be a quicker, cleverer way to create the SUBS array, which will take up a lot of time.

```
PRO KIM_SUBSCRIPT

ARR_2D = FINDGEN(9,10)        ; Input 2D array
ARR_1D = FLTARR(18)           ; Output 1D array (bins)

; MAKE UP SUBSCRIPTS FOR THE ARRAY TO FIT INTO THE BINS (KNOWN BIN
LOCATIONS)
SUBS = LONG(ARR_2D)
S = [REVERSE(INDGEN(9)+1),INDGEN(9)+1]
FOR N=0, N_ELEMENTS(S)-1 DO BEGIN
  IF N EQ 0 THEN FSUB = 0 ELSE FSUB = LSUB
  IF N EQ 0 THEN LSUB = S(N)-1 ELSE LSUB = FSUB + S(N)
  SUBS(FSUB:LSUB) = N
ENDFOR


; LOOP METHOD
FOR N=0, N_ELEMENTS(ARR_1D)-1 DO BEGIN
  OK = WHERE(SUBS EQ N, COUNT)
  IF COUNT GE 1 THEN ARR_1D(N) = MEAN(ARR_2D(OK))
ENDFOR
```

```
PRINT, 'ARR_2D'
PRINT, BYTE(ARR_2D)
PRINT
PRINT, 'S'
PRINT, BYTE(S)
PRINT
PRINT, 'SUBS'
PRINT, BYTE(SUBS)
PRINT
PRINT, 'ARR_1D'
PRINT, ARR_1D

; SLIGHTLY-QUICKER LOOP METHOD (TRUST THE SUBS ARRAY)
FOR N=0, N_ELEMENTS(ARR_1D)-1 DO ARR_1D(N) = MEAN(ARR_2D(WHERE(SUBS EQ
N)))


; HISTOGRAM METHOD

ARR_1D_H = FLTARR(18)            ; Output 1D array (bins)

H = HISTOGRAM(SUBS, MIN=0, REVERSE_INDICES=R)
FOR N=0, N_ELEMENTS(ARR_1D_H)-1 DO BEGIN
  COUNT = R[N+1]-R[N]
  IF COUNT GE 1 THEN ARR_1D_H(N) = MEAN(ARR_2D(R[R[N]:R[N+1]-1]))
ENDFOR

PRINT
PRINT, 'ARR_1D_H'
PRINT, ARR_1D_H

; SLIGHTLY-QUICKER HISTOGRAM METHOD (TRUST THE SUBS ARRAY)

ARR_1D_H = FLTARR(18)            ; Output 1D array (bins)

H = HISTOGRAM(SUBS, MIN=0, REVERSE_INDICES=R)
FOR N=0, N_ELEMENTS(ARR_1D_H)-1 DO BEGIN
  ARR_1D_H(N) = MEAN(ARR_2D(R[R[N]:R[N+1]-1]))
ENDFOR

PRINT
PRINT, 'ARR_1D_H'
PRINT, ARR_1D_H

END
```

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada --- http://www.d-jackson.com

Subject: Re: Subscripting help
Posted by Jeremy Bailin on Thu, 29 Oct 2015 18:28:23 GMT
View Forum Message <> Reply to Message

On Thursday, October 29, 2015 at 10:44:28 AM UTC-4, KH wrote:
> Hello,
>
> I need to convert a large 2D array to a smaller 1D array by taking the mean of several pixels in the 2D array before putting them into known locations in the 1D array (or bin). The challenge is that the number of pixels per bin varies so I don't know how to do it without creating a large loop. I was hoping there was a way I could do it with subscripts, but I haven't been able to figure it out yet.
>
> Here is an example of what I want to do using a loop. It works fine for this example, but my real 2D array is 8640x4320 and I need to move it to a ~5.6 million 1D array and it would take days to run it in a loop.
>
> ARR_2D = FINDGEN(9,10) ; Input 2D array
> ARR_1D = FLTARR(18) ; Output 1D array (bins)
>
> ; MAKE UP SUBSCRIPTS FOR THE ARRAY TO FIT INTO THE BINS (KNOWN BIN LOCATIONS)
> SUBS = LONG(ARR_2D)
> S = [REVERSE(INDGEN(9)+1),INDGEN(9)+1]
> FOR N=0, N_ELEMENTS(S)-1 DO BEGIN
> IF N EQ 0 THEN FSUB = 0 ELSE FSUB = LSUB
> IF N EQ 0 THEN LSUB = S(N)-1 ELSE LSUB = FSUB + S(N)
> SUBS(FSUB:LSUB) = N
> ENDFOR
>
>
> ; LOOP METHOD
> FOR N=0, N_ELEMENTS(ARR_1D)-1 DO BEGIN
> OK = WHERE(SUBS EQ N, COUNT)
> IF COUNT GE 1 THEN ARR_1D(N) = MEAN(ARR_2D(OK))
> ENDFOR
>
> PRINT, ARR_2D
> PRINT
> PRINT, ARR_1D
>

> Is there a faster way to get the same results? I was trying to figure out a way to do it with subscripts, but haven't had much success.
>
> Thanks for your input.
> Kim

This article is must-read for any problem like this. Basically, you want to use the double-histogram technique.

http://www.idlcoyote.com/code_tips/drizzling.html

-Jeremy.

Subject: Re: Subscripting help
Posted by Russell[1] on Thu, 29 Oct 2015 20:34:09 GMT
View Forum Message <> Reply to Message

WOW... Again, Jeremy... I logged on, saw the question and there's my answer already...

This does sound an awful lot like a drizzle type thing... I'm currently working on a project like this, and I found that using the array decimation techniques along with Jeremy's ideas of "index compression" (ie. his stuff with value_locate) to be indispensable in making these calculations work on a human timescale (and even that they're still pretty slow). You are using the "mean", so if you use the array decimation stuff that is on Fanning's website that is essentially the numerator of an average... So you'll just need to keep track of the number of points that are indexed to ARR_2D. Stemming from Jeremy's stuff and a number of posts by people on Fanning's webpage, I've written my own version of all sorts of stuff:

n_uniq()
uniqify()
decimate_array()
compress_indices()
sparse_histogram()

Good luck, this is a tough nut to crack, but when you get it--- it's super cool.
Russell

On Thursday, October 29, 2015 at 10:44:28 AM UTC-4, KH wrote:
> Hello,
>
> I need to convert a large 2D array to a smaller 1D array by taking the mean of several pixels in the 2D array before putting them into known locations in the 1D array (or bin). The challenge is that the number of pixels per bin varies so I don't know how to do it without creating a large loop. I was hoping there was a way I could do it with subscripts, but I haven't been able to figure it out

yet.
>
> Here is an example of what I want to do using a loop. It works fine for this example, but my real 2D array is 8640x4320 and I need to move it to a ~5.6 million 1D array and it would take days to run it in a loop.
>
> ARR_2D = FINDGEN(9,10) ; Input 2D array
> ARR_1D = FLTARR(18) ; Output 1D array (bins)
>
> ; MAKE UP SUBSCRIPTS FOR THE ARRAY TO FIT INTO THE BINS (KNOWN BIN LOCATIONS)
> SUBS = LONG(ARR_2D)
> S = [REVERSE(INDGEN(9)+1),INDGEN(9)+1]
> FOR N=0, N_ELEMENTS(S)-1 DO BEGIN
> IF N EQ 0 THEN FSUB = 0 ELSE FSUB = LSUB
> IF N EQ 0 THEN LSUB = S(N)-1 ELSE LSUB = FSUB + S(N)
> SUBS(FSUB:LSUB) = N
> ENDFOR
>
>
> ; LOOP METHOD
> FOR N=0, N_ELEMENTS(ARR_1D)-1 DO BEGIN
> OK = WHERE(SUBS EQ N, COUNT)
> IF COUNT GE 1 THEN ARR_1D(N) = MEAN(ARR_2D(OK))
> ENDFOR
>
> PRINT, ARR_2D
> PRINT
> PRINT, ARR_1D
>
> Is there a faster way to get the same results? I was trying to figure out a way to do it with subscripts, but haven't had much success.
>
> Thanks for your input.
> Kim

Subject: Re: Subscripting help
Posted by Jeremy Bailin on Fri, 30 Oct 2015 18:54:11 GMT
View Forum Message <> Reply to Message

On Thursday, October 29, 2015 at 4:34:12 PM UTC-4, rrya...@gmail.com wrote:
> WOW... Again, Jeremy... I logged on, saw the question and there's my answer already...
>
>
> This does sound an awful lot like a drizzle type thing... I'm currently working on a project like this, and I found that using the array decimation techniques along with Jeremy's ideas of "index compression" (ie. his stuff with value_locate) to be indispensable in making these calculations

work on a human timescale (and even that they're still pretty slow). You are using the "mean", so if you use the array decimation stuff that is on Fanning's website that is essentially the numerator of an average... So you'll just need to keep track of the number of points that are indexed to ARR_2D. Stemming from Jeremy's stuff and a number of posts by people on Fanning's webpage, I've written my own version of all sorts of stuff:
>
> n_uniq()
> uniqify()
> decimate_array()
> compress_indices()
> sparse_histogram()
>
> Good luck, this is a tough nut to crack, but when you get it--- it's super cool.
> Russell
>
>
>
> On Thursday, October 29, 2015 at 10:44:28 AM UTC-4, KH wrote:
>> Hello,
>>
>> I need to convert a large 2D array to a smaller 1D array by taking the mean of several pixels in the 2D array before putting them into known locations in the 1D array (or bin). The challenge is that the number of pixels per bin varies so I don't know how to do it without creating a large loop. I was hoping there was a way I could do it with subscripts, but I haven't been able to figure it out yet.
>>
>> Here is an example of what I want to do using a loop. It works fine for this example, but my real 2D array is 8640x4320 and I need to move it to a ~5.6 million 1D array and it would take days to run it in a loop.
>>
>> ARR_2D = FINDGEN(9,10) ; Input 2D array
>> ARR_1D = FLTARR(18) ; Output 1D array (bins)
>>
>> ; MAKE UP SUBSCRIPTS FOR THE ARRAY TO FIT INTO THE BINS (KNOWN BIN LOCATIONS)
>> SUBS = LONG(ARR_2D)
>> S = [REVERSE(INDGEN(9)+1),INDGEN(9)+1]
>> FOR N=0, N_ELEMENTS(S)-1 DO BEGIN
>> IF N EQ 0 THEN FSUB = 0 ELSE FSUB = LSUB
>> IF N EQ 0 THEN LSUB = S(N)-1 ELSE LSUB = FSUB + S(N)
>> SUBS(FSUB:LSUB) = N
>> ENDFOR
>>
>>
>> ; LOOP METHOD
>> FOR N=0, N_ELEMENTS(ARR_1D)-1 DO BEGIN
>> OK = WHERE(SUBS EQ N, COUNT)
>> IF COUNT GE 1 THEN ARR_1D(N) = MEAN(ARR_2D(OK))

>> ENDFOR
>>
>> PRINT, ARR_2D
>> PRINT
>> PRINT, ARR_1D
>>
>> Is there a faster way to get the same results? I was trying to figure out a way to do it with subscripts, but haven't had much success.
>>
>> Thanks for your input.
>> Kim

I blush... :)

If you want to see some really edge-of-your-pants, very efficient, and poorly documented use of the double histogram technique using value_locate to deal with sparse histograms, you can look at MATCH_ND (most of the hard work is in MATCHALL_ND), or if you feel really brave MATCH_SPH/MATCHALL_SPH in JBIU:

http://www.simulated-galaxies.ua.edu/jbiu/

Cheers,
-Jeremy.