Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by Michael Galloy on Tue, 23 May 2006 00:27:20 GMT
View Forum Message <> Reply to Message

Jonathan Wolfe wrote:
> Hello,
>
> I have been trying to average components of a time series when they
> meet certain criteria.
> For example,
>
> I have values in which I use a where statement to specify the critera
>
> criteria=where(time lt 1000)
>
> and I get an array returned which looks like this
>
> data=[0,1,2,3,4,6,8,9,10]
>
> Now, given this array, I would like to specify individual arrays for
> any "block" of subsequent numbers with size greater than three.
>
> for this array it would look like this
>
> a=[0,1,2,3,4]
> b=[8,9,10]
>
> I tried using the complement keyword in the where statement to put the
> null values into an array and try to subscript my way through the
> answer
>
> where(-------,complement=q)
>
> result=data[0:Q(0)-1]
> result1=[Q(0):Q(1)]
>
> that turned out to be quite messy especially since I am using multiple
> files which all have a different "patterns of three or more"
>
> I've messed around with for loops and if statements, but again I have
> to change those for each individual file. It would be nice to know a
> technique which could accomplish what I am trying to do:
>
> find series of subsequent integers in an array to make multiple new
> arrays.
>
>

> This seems like a simple problem, but I haven't been able to figure it
> out. Maybe I'm overlooking something... Any help would be appreciated!
>
> Thank you in advance!
> Jon
>

I think something like this might work for you:

```
; I'm just setting up a fake data set like you have
n = 11
time = fltarr(n)
time[[5, 7]] = 1000

; morphological logical operators to eliminate groups smaller than three
k = [1, 1, 1]
r = dilate(erode(time lt 1000, k), k)

; label individual groups
regions = label_region([0, r, 0])
regions = regions[1:n]

; go through each group and print (or do whatever)
h = histogram(regions, reverse_indices=ri)
for i = 1L, n_elements(h) - 1L do print, ri[ri[i]:ri[i+1]-1
```

Mike
--
www.michaelgalloy.com

Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by JD Smith on Tue, 23 May 2006 01:19:15 GMT
View Forum Message <> Reply to Message

On Mon, 22 May 2006 15:10:08 -0700, Jonathan Wolfe wrote:

> Hello,
>
> I have been trying to average components of a time series when they meet
> certain criteria.
> For example,
>
> I have values in which I use a where statement to specify the critera
>
> criteria=where(time lt 1000)
>

> and I get an array returned which looks like this
>
> data=[0,1,2,3,4,6,8,9,10]
>
> Now, given this array, I would like to specify individual arrays for any
> "block" of subsequent numbers with size greater than three.

It's actually not as simple as it seems. Here's a method which uses
LABEL_REGIONS and HISTOGRAM:

```
l=label_region([0L,(shift(data,-1)-data) eq 1,0L])
h=histogram(l[1:n_elements(l)-2],MIN=1,REVERSE_INDICES=ri)
wh=where(h ge 3-1,cnt)
for i=0,cnt-1 do print,data[ri[ri[wh[i]]]:ri[ri[wh[i]+1]-1]+1]
```

Note that LABEL_REGIONS is annoying in that it calls the end point as
"no region", so we must temporarily surround it with buffer 0's.
Careful of my usage of REVERSE_INDICES there... instead of the normal
semantic:

```
 data[ri[ri[i]:ri[i+1]-1]]
```

which is like data[index_vector], I instead use an explicit range:

```
 data[ri[r[i]]:ri[ri[i+1]-1]+1]
```

which is like data[low:high]. Stare at it until you see the
difference. I did this because I knew that the elements in the
labeled regions had consecutive indices, and because due to the SHIFT
call, we're always missing one member of the "consecutive run" at the
end (hence the +1). In general the indices in a given HISTOGRAM bucket
are not adjacent in the original array, so the first form is correct.

If you want to save these arrays as you go, perhaps a pointer array
would be useful.

JD

Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by vorticitywolfe on Tue, 23 May 2006 22:11:39 GMT
View Forum Message <> Reply to Message

Thank you both for your help! In regards to JD's saving the output
arrays with pointers... I have never used pointers and believe this is
a case in which they are necessary.

After writing a long, drawn out explanation of where I was stuck with
pointers I ended up figuring them out. Just in case anyone new to
pointers wants to know how to get variables from an example such as the
above threads, use something like this:

```
x= ptrarr(3)

for i = 1L, n_elements(h) - 1L do begin
     t=ri[ri[i]:ri[i+1]-1]
     x[i] = PTR_NEW( t,/allocate_heap )
endfor

print,*x(2)
```

and you will have your varying size arrays of a larger array segmented
into different subscripts of x.

I'm sure there may be a better way to do this, but it makes sense to me.

Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by Paul Van Delst[1] on Tue, 23 May 2006 22:26:38 GMT

View Forum Message <> Reply to Message

Jonathan Wolfe wrote:

```
>  Thank you both for your help!  In regards to JD's saving the output
>  arrays with pointers... I have never used pointers and believe this is
>  a case in which they are necessary.
>
>  After writing a long, drawn out explanation of where I was stuck with
>  pointers I ended up figuring them out.  Just in case anyone new to
>  pointers wants to know how to get variables from an example such as the
>  above threads, use something like this:
>
>  x= ptrarr(3)
>
>  for i = 1L, n_elements(h) - 1L do begin
>       t=ri[ri[i]:ri[i+1]-1]
>       x[i] = PTR_NEW( t,/allocate_heap )
>  endfor
>
>  print,*x(2)
>
>  and you will have your varying size arrays of a larger array segmented
>  into different subscripts of x.
>
>  I'm sure there may be a better way to do this, but it makes sense to me.
```

Not a big change, but you can also do:

```
  n=n_elements(h)
  x=ptrarr(n,/ALLOCATE_HEAP)
  for i = 0L, n-1L do begin
        t=ri[ri[i]:ri[i+1]-1]
        *x[i] = t
  endfor
```

BTW, do you really want
```
   for i = 1L, n_elements(h) - 1L
```
or
```
   for i = 0L, n_elements(h) - 1L
```

??

You never use x[0] in your orig code.

paulv

--
Paul van Delst   Ride lots.
CIMSS @ NOAA/NCEP/EMC   Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by David Fanning on Tue, 23 May 2006 22:55:42 GMT
View Forum Message <> Reply to Message

Jonathan Wolfe writes:

```
>  Thank you both for your help!  In regards to JD's saving the output
>  arrays with pointers... I have never used pointers and believe this is
>  a case in which they are necessary.
>
>  After writing a long, drawn out explanation of where I was stuck with
>  pointers I ended up figuring them out.  Just in case anyone new to
>  pointers wants to know how to get variables from an example such as the
>  above threads, use something like this:
>
>  x= ptrarr(3)
>
>  for i = 1L, n_elements(h) - 1L do begin
>        t=ri[ri[i]:ri[i+1]-1]
>       x[i] = PTR_NEW( t,/allocate_heap )
>  endfor
>
>  print,*x(2)
>
```
> and you will have your varying size arrays of a larger array segmented
> into different subscripts of x.
>
> I'm sure there may be a better way to do this, but it makes sense to me.

I wrote up a slightly different method on my web page:

http://www.dfanning.com/idl_way/avgseries.html

Be sure you free up your pointers when you are finished with them:

```
  Ptr_Free, x
```

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by JD Smith on Wed, 24 May 2006 21:55:59 GMT
View Forum Message <> Reply to Message

On Tue, 23 May 2006 16:55:42 -0600, David Fanning wrote:

> Jonathan Wolfe writes:

> I wrote up a slightly different method on my web page:
>
>   http://www.dfanning.com/idl_way/avgseries.html

Some remixing there... mostly fine, but at one point you you say
"although not in this particular case" regarding the correctness of
the form of reverse indices indexing. That's actually not true: both
forms are correct, it's just that with the data[index_vector] form,
it's not easy to extend the range of indices by one. You could
actually say:

```
 data[[index_vector,index_vector[n_elements(index_vector)-1]] ]
```

and this would do it just as well. Using the data[low:high] trick,
where you know you have consecutive indices in your HISTOGRAM bin,
just allows you to append one to the range of indices to recover that
mis sing member of the group. So both are correct, but one is more
convienent.

JD

Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by David Fanning on Wed, 24 May 2006 22:23:52 GMT

View Forum Message <> Reply to Message

JD Smith writes:

> Some remixing there... mostly fine, but at one point you you say
> "although not in this particular case" regarding the correctness of
> the form of reverse indices indexing. That's actually not true: both
> forms are correct, it's just that with the data[index_vector] form,
> it's not easy to extend the range of indices by one. You could
> actually say:
>
> data[[index_vector,index_vector[n_elements(index_vector)-1]]]
>
> and this would do it just as well. Using the data[low:high] trick,
> where you know you have consecutive indices in your HISTOGRAM bin,
> just allows you to append one to the range of indices to recover that
> mis sing member of the group. So both are correct, but one is more
> convienent.

OK, I don't honestly know what it means either way, but I
fixed it to indicate both are correct. :-)

Cheers,

David

P.S. Let's just say if both are correct, the solution is even
MORE of a mystery to me! And I like it that way.
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by JD Smith on Wed, 24 May 2006 22:59:40 GMT
View Forum Message <> Reply to Message

On Wed, 24 May 2006 16:23:52 -0600, David Fanning wrote:

> JD Smith writes:
>
>> Some remixing there... mostly fine, but at one point you you say
>> "although not in this particular case" regarding the correctness of
>> the form of reverse indices indexing. That's actually not true: both
>> forms are correct, it's just that with the data[index_vector] form,
>> it's not easy to extend the range of indices by one. You could
>> actually say:
>>
>> data[[index_vector,index_vector[n_elements(index_vector)-1]]]
>>
>> and this would do it just as well. Using the data[low:high] trick,
>> where you know you have consecutive indices in your HISTOGRAM bin,
>> just allows you to append one to the range of indices to recover that
>> mis sing member of the group. So both are correct, but one is more
>> convienent.
>
> OK, I don't honestly know what it means either way, but I
> fixed it to indicate both are correct. :-)

It's just the difference between data[[1,2,3,4,5]] and data[1:5]. No big
deal. If you want to add "6", it's easier to use the second form than the
first.

JD

Subject: Re: Create new arrays from series of subsequent integers in an existing array
Posted by David Fanning on Wed, 24 May 2006 23:04:43 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> It's just the difference between data[[1,2,3,4,5]] and data[1:5]. No big
> deal. If you want to add "6", it's easier to use the second form than the
> first.

Ah, now *that* I understand! :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/