Subject: Indexing arrays with arrays
Posted by Conor on Fri, 21 Sep 2007 15:18:38 GMT
View Forum Message <> Reply to Message

A couple times I've asked about this. I sometimes would like to be able to index arrays with arrays without using for loops. For intance imagine I have:

```
data = findgen(20)

st = [5,2,10]
ed = [6,4,15]
```

I would like to do:

```
res = data[st:ed]
```

obviously, IDL doesn't allow this. The last time I posted about this, I got this response:

Well, if end_ind is always a constant offset from start_ind, it's easy just to contruct the index vector yourself:

```
 t=[end_ind[0]-start_ind[0]+1,n_elements(start_ind)]
 extract=res[reform(rebin(1#start_ind,t)
+rebin(indgen(t[0]),t),t[0]*t[1])]
```

If the start-end difference can be any variable amount, this is a problem for HISTOGRAM related to chunk indexing. See the HISTOGRAM tutorial

This was very helpful and got me started. After much pondering, I came up with this for the general case. Assume that st and ed are row arrays (as per the above example)

```
function array_index,st,ed

nst = n_elements(st)
ned = n_elements(ed)
diff = ed - st + 1

h = histogram(total(diff,/cumulative)-1,/
binsize,min=0,reverse_indices=ri)
i = ri[0:n_elements(h)-1]-ri[0]

arr = [fltarr(1,nst),transpose(diff)-1]
maxdiff = max(diff)
x = rebin(findgen(maxdiff),maxdiff,nst)/(maxdiff-1)
```

```
y = rebin(findgen(1,nst),maxdiff,nst)
int = interpolate(arr,x,y)
ref = reform(ceil(int),maxdiff*nst)
adds = ref[uniq(ref)]

return,st[i]+adds
```

Given the above example:

```
data = indgen(20)
st = [5,2,10]
ed = [6,4,15]

print,data[array_index(st,ed)]

; prints       5       6       2       3       4      10      11
12      13      14      15
```

I'm rather happy about the result. There's been a couple times when I really wish I had this available. Anyway, a couple questions:

1) Anyone see a better way to do this?
2) Anyone want to generalize this to n dimensions?
i.e. inds = array_index([[st1,ed1],[st2,ed2],[st3,ed3]])

Subject: Re: indexing
Posted by Jeremy Bailin on Thu, 02 Apr 2009 01:17:23 GMT
View Forum Message <> Reply to Message

On Apr 1, 9:12 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
> I swear I've seen this come up within the past few months, but I can't
> find it, so here goes:
>
> Let's say I have a 3D array - think of it as x,y,z if you want. I have
> a list of x,y pairs and I want to perform on operation on a given
> range in z (say z1:z2) and each x,y pair. For a simple example, let's
> say I just want them all up. So, if the following were my x,y pairs:
> 3 4
> 2 1
> 3 7
> and my z range was 1:3 then I want
> array[3,4,1]+array[3,4,2]+array[3,4,3]
> +array[2,1,1]+array[2,1,2]+array[2,1,3]
> +array[3,7,1]+array[3,7,2]+array[3,7,3]
>
> Is there a simple representation for this? My standard solution, if

> pair is an npair x 2 array containing the x,y pairs, looks something
> like this:
> nz=z2-z1+1
> zindices=rebin(reform(z1+lindgen(nz),1,nz), npair,nz)
> xindices=rebin(pair[*,0],npair,nz)
> yindices=rebin(pair[*,1],npair,nz)
> answer = total(array[xindices,yindices,zindices])
>
> ...but if nz and npair are large, generating all of those 2D index
> arrays is really wasteful. The following also works:
>
> answer = total(array[pair[*,0],pair[*,1],z1:z2] * rebin(identity
> (npair,npair,nz)))
>
> but again generates 2 intermediate npair x npair x nz arrays that are
> wasteful if npair is large.
>
> Any takers?
>
> -Jeremy.

...that last line should read:

answer = total(array[pair[*,0],pair[*,1],z1:z2] * rebin(identity
(npair),npair,npair,nz))

-Jeremy.

Subject: Re: indexing
Posted by Chris[6] on Thu, 02 Apr 2009 05:34:24 GMT
View Forum Message <> Reply to Message

I would have done something like this, accomplishing the summation in
two steps

x = [3, 2, 3]
y = [4, 1, 7]
sum = total(array[*,*,1:3], 3) ; summed along z direction
answer = total(sum[x,y]) ; summed over x,y pairs

chris

Subject: Re: indexing
Posted by Jeremy Bailin on Thu, 02 Apr 2009 12:25:56 GMT

On Apr 2, 1:34 am, Chris <beaum...@ifa.hawaii.edu> wrote:
> I would have done something like this, accomplishing the summation in
> two steps
>
> x = [3, 2, 3]
> y = [4, 1, 7]
> sum = total(array[*,*,1:3], 3) ; summed along z direction
> answer = total(sum[x,y]) ; summed over x,y pairs
>
> chris

Yeah, that definitely works nicely for total (I like the fact that it doesn't need any internal index vectors, and that the intermediate stage is a constant size independent of npair and nz), which is what I'm doing today. But it wouldn't work for the more generic case where I want to do something else to those values - for example, I've needed to do the equivalent with median before, and that won't work as a 2-step process.

-Jeremy.

Subject: Re: indexing
Posted by Dick Jackson on Thu, 02 Apr 2009 16:49:20 GMT

Hi all,

Good one, Chris. How about this for getting the subset:

```
PRO IndexXYPairsOverZ

a = Transpose(IndGen(10,10,10), [2,1,0]) ; Array where, e.g. a[3,0,4]=304

xy = [[3,4],[2,1],[3,7]]
z0 = 1
z1 = 3

dims=Size(a,/Dim)
a = Reform(a, dims[0]*dims[1], dims[2], /Overwrite) ; Reshape a temporarily
xyIndices = xy[0,*]+xy[1,*]*dims[0]
subset = a[xyIndices, z0:z1]
a = Reform(a, dims, /Overwrite)                    ; Restore a's shape

Help, subset
Print, subset
```

END

Printed result:

```
SUBSET          INT       = Array[3, 3]
    341     211     371
    342     212     372
    343     213     373
```

Note that the Reforms will take effectively no time at all, and you only make one list of indices.

Hope this helps.

Cheers,
-Dick

--
Dick Jackson Software Consulting http://www.d-jackson.com
Victoria, BC, Canada +1-250-220-6117 dick@d-jackson.com

"Jeremy Bailin" <astroconst@gmail.com> wrote in message
news:a50cae0f-63e5-47b7-a44e-b5363114855a@r37g2000yqn.googlegroups.com...
On Apr 2, 1:34 am, Chris <beaum...@ifa.hawaii.edu> wrote:
> I would have done something like this, accomplishing the summation in
> two steps
>
> x = [3, 2, 3]
> y = [4, 1, 7]
> sum = total(array[*,*,1:3], 3) ; summed along z direction
> answer = total(sum[x,y]) ; summed over x,y pairs
>
> chris

Yeah, that definitely works nicely for total (I like the fact that it doesn't need any internal index vectors, and that the intermediate stage is a constant size independent of npair and nz), which is what I'm doing today. But it wouldn't work for the more generic case where I want to do something else to those values - for example, I've needed to do the equivalent with median before, and that won't work as a 2-step process.

-Jeremy.