Subject: Re: All twisted up over reverse indices!
Posted by PMan on Fri, 16 Dec 2016 13:37:02 GMT
View Forum Message <> Reply to Message

On Friday, December 16, 2016 at 5:51:42 AM UTC-5, Stephen Williams wrote:
> Greetings all,
>
> I'm slowly working my way through enlightenment with the wonderful "matchall_sph.pro" from Jeremy Bailin.
>
> However, I think I've made a terrible misinterpretation with the indices, and I'm hoping for clarification.
>
> My problem: I have two catalogs of positions for stars that I'm trying to match. I'm also interested in multiple matches and distances between said matches, all of which can be extracted from the reverse indices and other output of "matchall_sph.pro". However, I'm a bit confused as to how to do the proper notation for recovery. An idea of my code thus far:
>
> ---
>
> match1 = matchall_sph(r_bv,d_bv,ra_i,de_i,search_rad,nwithin,distance =d)
>
> part_one = where(nwithin gt 0) ; Indices of matched BV sources
> part_two = where(nwithin eq 1) ; same, but only singly matched sources
> part_thr = where(nwithin gt 1) ; same, but only multiply matched sources
>
> ;
> ; This should extract the RA and Dec of singly matched sources.
> ;
>
> ra_i_m = dblarr(n_elements(part_two))
> de_i_m = dblarr(n_elements(part_two))
>
> for i=0L,n_elements(part_two)-2 do begin
>
> ra_i_m(i) = ra_i(match1[match1[part_two(i)]:match1[part_two(i+1)]-1])
> de_i_m(i) = de_i(match1[match1[part_two(i)]:match1[part_two(i+1)]-1])
>
> endfor
>
> ---
>
> Now, I'm pretty sure this worked like a charm, as a later output file contains the positions from the BV catalog and I catalog, and they make sense, given my inputs. (I also had to write a one-line command to get the final value from that loop, because the "i+1" wrecked the loop!! Yes, I'm that new!)
>
> Not for want of trying, but I can't wrap my mind around extracting the multiple sources, nor the

distances from those sources. For example, I've tried the following: (just to see the multiple matches from this particular index)
>
> ----------
>
> r_tmp = ra_i(match1[match1[part_thr(0)]:match1[part_thr(1)]-1])
>
> print,n_elements(r_tmp)
>
> ----------
>
> This results in 250 elements, when I know that the number of matched I-band sources to this one BV source is only 2. What subscripting pitfall have I stumbled upon? Or is referencing "match1" with the array "part_thr" not appropriate here?
>
> And how would I extract distances given the following in the help file:
>
> ----------
>
> ; DISTANCE: Optional output containing the distances between each ;pair.
> ; The distances are stored in the same order as the Result
> ; array but starting at 0, i.e. if j is match number k to
> ; element i then
> ; j = Result[Result[i]+k]
> ; and the distance between points i and j is
> ; DISTANCE[Result[i]+k-Result[0]]
>
> ----------
>
> Thanks tons to anyone who can help, with any part of this mystery! My brain hurts at this point!
>
> Stephen

I found this helped (scroll down to the section on reverse indices):
http://www.idlcoyote.com/tips/histogram_tutorial.html

Subject: Re: All twisted up over reverse indices!
Posted by Stephen Williams on Mon, 19 Dec 2016 17:37:42 GMT
View Forum Message <> Reply to Message

>
> I found this helped (scroll down to the section on reverse indices):
> http://www.idlcoyote.com/tips/histogram_tutorial.html

For completeness, I figured out the problem and determined how to extract what I needed. My issue came primarily from improper referencing of the arrays. For example, the following line gave the incorrect indices when used with multiple matches:

ra_i_m(i) = ra_i(match1[match1[part_two(i)]:match1[part_two(i+1)]-1])

The problem is in the end, and should be this:

ra_i_m(i) = ra_i(match1[match1[part_two(i)]:match1[part_two(i)+1]-1])

The parentheses are critical here, as I want the next array element in "match1" and not the next array element in "part_two".

Distances were obtained similarly via the following line:

d(match1[match1[part_thr(i)]:match1[part_thr(i)+1]-1])

That is to say, the above gives the distances for each element in the sub-array denoted by the ":".

I hope this may help someone, I am sure relieved that my problem has been solved.

Subject: Re: All twisted up over reverse indices!
Posted by penteado on Mon, 19 Dec 2016 20:45:36 GMT
View Forum Message <> Reply to Message

In case anyone is wondering, the reverse indices can be converted into lists or hashes, which make their use much simpler, with

```
 h=histogram(array,_strict_extra=_ex,reverse_indices=ri,locat ions=locations)
if arg_present(rl) then begin
  rl=list(length=n_elements(h))
  foreach el,h,i do if (el gt 0L) then rl[i]=ri[ri[i]:ri[i+1]-1]
endif
if arg_present(rh) then begin
  rh=!version.release ge '8.3' ? orderedhash(locations) : hash(locations)
  foreach el,h,i do if (el gt 0L) then rh[locations[i]]=ri[ri[i]:ri[i+1]-1]
endif
```

This is used in my histogram wrapper, histogram_pp (http://www.ppenteado.net/idl/pp_lib/doc/histogram_pp.html):

```
 vals=randomu(0L,15)*13d0
  h=histogram_pp(vals,reverse_indices=ri,reverse_list=rl,locat ions=loc,reverse_hash=rh)
 print,h
 ;1       3       0       4       2       1       0       4
 print,ri[ri[0]:ri[1]-1]
 ;13
 foreach el,rl,i do print,i,' : ',el
 ;0 :       13
 ;1 :        8       11       12
```

```
;2 : !NULL
;3 :           0          1          4          6
;4 :           9         10
;5 :           2
;6 : !NULL
;7 :           3          5          7         14
foreach el,loc do print,el,' : ',rh[el]
;3.8679500 :          13
;4.8679500 :           8         11         12
;5.8679500 : !NULL
;6.8679500 :           0          1          4          6
;7.8679500 :           9         10
;8.8679500 :           2
;9.8679500 : !NULL
;10.867950 :           3          5          7         14
```

On Monday, December 19, 2016 at 9:37:46 AM UTC-8, Stephen Williams wrote:
>>
>> I found this helped (scroll down to the section on reverse indices):
>> http://www.idlcoyote.com/tips/histogram_tutorial.html
>
> For completeness, I figured out the problem and determined how to extract what I needed. My issue came primarily from improper referencing of the arrays. For example, the following line gave the incorrect indices when used with multiple matches:
>
> ra_i_m(i) = ra_i(match1[match1[part_two(i)]:match1[part_two(i+1)]-1])
>
> The problem is in the end, and should be this:
>
> ra_i_m(i) = ra_i(match1[match1[part_two(i)]:match1[part_two(i)+1]-1])
>
> The parentheses are critical here, as I want the next array element in "match1" and not the next array element in "part_two".
>
> Distances were obtained similarly via the following line:
>
> d(match1[match1[part_thr(i)]:match1[part_thr(i)+1]-1])
>
> That is to say, the above gives the distances for each element in the sub-array denoted by the ":".
>
> I hope this may help someone, I am sure relieved that my problem has been solved.