Subject: Re: how to loop through data?
Posted by ben.bighair on Thu, 03 Feb 2011 00:59:49 GMT
View Forum Message <> Reply to Message

On 2/2/11 6:03 PM, smuzz wrote:
> On Feb 2, 5:42 pm, Ammar Yusuf<amyus...@gmail.com> wrote:
>> On Feb 2, 5:29 pm, smuzz<smussol...@gmail.com> wrote:
>>
>>
>>
>>> Hi --
>>
>>> I am new to programming in IDL and I want to execute a for loop. The
>>> data I have contains number of whale calls from different species over
>>> a 2 month period. I have manually gone through and specified the start
>>> and end of periods of calling that I want to focus on (i.e. bouts).
>>> Now I want to look 1 hour before the start of each bout and 1 after
>>> the end of each bout to total up all the whale calls that I hear from
>>> the different species. I have about 20 bouts and ideally would like to
>>> write some sort of for loop that will go through each bout and tally
>>> up calls, but I am a bit confused how I would go about this since my
>>> bouts are all different and appreciate any suggestions...
>>
>>> ***********************
>>> What I have so far:
>>
>>> ; This finds all the calls for the 1st species
>>> l = where(auto(*).species eq 7 and auto(*).call_type eq 1 and
>>> auto(*).start_time gt stime and auto(*).start_time lt etime,lcount)
>>
>>> ; This finds all the calls for the 2nd species
>>> m = where(auto(*).species eq 7 and auto(*).call_type eq 3 and
>>> auto(*).start_time gt stime and auto(*).start_time lt etime,mcount)
>>
>>> ; This finds all the calls for the 3rd species
>>> n = where(auto(*).species eq 10 and auto(*).call_type eq 0 and
>>> auto(*).start_time gt stime and auto(*).end_time lt etime,ncount)
>>
>>> ; These are the start and end times of a few of my bouts (I've already
>>> taken care of converting time)
>>> start_bout1 = '11/19/09 22:30:00'
>>> end_bout1 = '11/19/09 23:21:00'
>>> start_bout2 = '11/20/09 02:06:00'
>>> end_bout2= '11/20/09 02:56:00'
>>> start_bout3= '11/20/09 19:27:00'
>>> end_bout3= '11/20/09 20:07:00'
>>> start_bout4= '11/20/09 21:37:00'
>>> end_bout4= '11/20/09 22:49:00'

>>> start_bout......
>>
>>> ********************
>>
>>> Thanks, S
>>
>> Hi, I'm not sure what exactly you're looking for but I'll give it a
>> shot.
>>
>> for i = 0, n_elements(l) do begin
>>   print, l[i]
>>   print, auto[l[i]]
>>   ; do some stuff with auto[l[i]] because I'm assuming that's the
>> index that you wanted.
>>   ; do some more stuff
>> endfor
>>
>> Not sure if this is what you were looking for.
>
> Thanks for the advice and I apologize for the confusion in my
> question. I guess my question is -- how does this for loop factor in
> looking 1 hour before and after the start and end of the bouts?

Hi,

Neato! Whale calls with IDL. Who would ever think it!

I'm not sure that a loop needs to be involved at all to identify the
calls within a bout. I think you can use VALUE_LOCATE to identify the
'calls' that fit with in 'bouts'.

Ignore for a moment that bouts aren't a span of time but instead an
instance. You can use VALUE_LOCATE to identify the call that just
proceeds a bout.

```
; first make up some call times - for my example they are evenly spaced
; days which isn't likely to be what you have
t0 = SYSTIME(/JULIAN)
calls = TIMEGEN(365, UNIT = "Days") + t0

; now make up some 'bouts'.  These I space 15 days apart and start them
; at some arbitrary day of the year
bouts = TIMEGEN(12, UNIT = "Days", STEP = 15, START = 1.5) + t0


; now use VALUE_LOCATE to find the indices in 'call' where you'll find
; 'bouts'
ix = VALUE_LOCATE(call, bouts)
```

```
print, ix
for i = 0L, N_ELEMENTS(ix) -1 do $
   PRINT, bouts[i], calls[ix[i]]


; now we go back to the real definition of 'bout' - as a span of time
; create a delta of +/-1 hour
dt = [-1.0, 1.0]/24.0

; now we transform the bouts into a 2 column array [begin, end]
; and add the deltas - so each row is your bout window
bouts = REBIN(reform(bouts,1,12),2,12,/SAMPLE)
bouts[0,*] = bouts[0,*] + dt[0]
bouts[1,*] = bouts[1,*] + dt[1]

; now you can use VALUE_LOCATE again
iy = VALUE_LOCATE(call, bouts)
print, iy
for i = 0L, N_ELEMENTS(iy) -1 do $
   PRINT, bouts[i], calls[iy[i]]
```

I think you'll find that once you put in your real data it will make more sense. Also, you don't have to have the bout ranges in a 2x12 array. You could just as easily run

```
iStart = VALUE_LOCATE(call, boutStarts)
iEnd = VALUE_LOCATE(call, boutEnds)
```

or whatever suits your needs. If you did this in R you would use the function findInterval() instead of VALUE_LOCATE().

I hope that helps - it sounds like a cool project.

CHeers,
Ben