Subject: Re: Reading and Plotting big txt. File
Posted by incognito.me on Fri, 03 Aug 2007 14:15:51 GMT
View Forum Message <> Reply to Message

On 3 Aug., 14:31, Conor <cmanc...@gmail.com> wrote:
> On Aug 3, 7:43 am, "incognito.me" <incognito...@gmx.de> wrote:
>
>
>
>
>
>> On 2 Aug., 19:27, Conor <cmanc...@gmail.com> wrote:
>
>>> On Aug 2, 12:55 pm, Conor <cmanc...@gmail.com> wrote:
>
>>>> The problem is your format statement. What's going on is that with a
>>>> format, IDL doesn't actually read columns. It is more of directions
>>>> where to find the data. In your case, you aren't telling it where the
>>>> spaces are, so it assumes that everything is a data column. If you
>>>> specify 10(a4), it is really reading:
>
>>>> aaaabbbbccccddddeeeffffgggghhhhiiiijjjj
>
>>>> where aaaa = column1, bbbb = column2, etc...
>
>>>> You need to give it the appropriate number of spaces, otherwise the
>>>> data get's all messed up. For example, apply the above "filter" to
>>>> the data below (from your file)
>
>>>> 7 -1848 -1792 -1718 -1678 -1638 -1576 -1517
>>>> -1446 -1372 -1322
>
>>>> The first four columns ' 7 ' are assigned to the first column in your
>>>> data array. The second four columns ' ' go to the second column in
>>>> your data array, etc.. In the end you get:
>
>>>> data = [7 ',' ',' -1','848 ',' -17','92 ',' -17','18 ',' -16']
>
>>>> (or something along those lines, anyway)
>
>>>> What you need to do is actually specify where the spaces are:
>
>>>> format = '(a2, 7x, a4, 2x, a4, 7(3x, a4))'
>
>>>> I don't think that's quite it, but it probably needs to be something
>>>> along those lines. I can't quite get it to work myself,
>>>> unfortunately. I wish someone better informed about formats would
>>>> join in the conversation here...

```
>
>>>  Okay, here's a solution.  I didn't want to have to go here, because it
>>>  is possibly the worst way to solve this problem, but since I can't
>>>  figure out the formats and no one else has any suggestions, we'll just
>>>  do it the "bad" way.  It's bad because it is not a general solution
>>>  (this will only work this one sort of file), it's worse because it is
>>>  really slow, and it is even worse because neither of us is going to
>>>  figure out what is wrong with what we've been trying.  Oh well.  The
>>>  plan is to manually parse the file.  Rather than relying on format
>>>  statements, I wrote a program that reads the file in line by line and
>>>  parses it according to rules I give it.  Specifically, this program
>>>  works by telling it where each column starts and how long each column
>>>  is.  There's a couple caveats with this program.  First, it should
>>>  only read actual data - you'll have to remove the header to run this
>>>  program on it (or, you can leave the header in and add a couple
>>>  generic readf statements right after opening the file to read out the
>>>  header data before entering the main program loop).  Anyway, here's
>>>  the program, and I've tested it succesfully on the above text file.
>>>  Also, you can download the source directly
here:http://astro.ufl.edu/~cmancone/pros/parse_bigfile.pro
>
>>>  function parse_bigfile,filename
>
>>>  openr,lun,filename,/get_lun
>
>>>  st = [0,9,16,24,32,40,48,56,64,72,80]
>>>  len = [2,5,5,5,5,5,5,5,5,5,5]
>>>  num = n_elements(len)
>
>>>  line = ''
>>>  data = intarr(num)
>
>>>  l = 0
>>>  while not( eof(lun) ) do begin
>
>>>          ; read in the line and see how long it is
>>>          readf,lun,line
>>>          data = intarr(num)
>>>          length = strlen(line)
>
>>>          for i=0,num-1 do begin
>>>                  ; if we've moved past the end of the line, we are done with this
>>>  line
>>>                  if st[i] gt length-1 or length eq 0 then break
>
>>>                  ; read and process the current element
>>>                  data[i] = float( strmid( line, st[i], len[i] ) )
>>>          endfor
```

>
>>>   ; if this is the first line, create our data result. Otherwise, just
>>> append the new data
>>>   if l eq 0 then result = data else result = [[result],[data]]
>
>>>   ; increment our line counter
>>>   ++l
>>> endwhile
>
>>> close,lun
>>> free_lun,lun
>
>>> return,result
>
>>> end
>
>>> Now, the biggest problem with something like this is that you have to
>>> specify where every column stars. For 1000 columns, this is not a
>>> simple task. What you will have to do is see what the repeating
>>> pattern is (hopefully there is one). So, if the above file is any
>>> indication, columns are always 5 characters long with 3 spaces in
>>> between. That means that you can initialize the start array to
>>> something like:
>
>>> st = findgen(1000)*8
>
>>> of course, it won't be exactly that. If I take the above file as a
>>> guide, it would be more like this:
>
>>> st = [0,9,findgen(1000)*8 + 16]
>>> len = fltarr(1002) + 5
>
>>> since the first two columns don't follow the same pattern as the rest
>>> of them. Just make sure that len and st have the same number of
>>> elements in them. Also, remember that starting positions for strings
>>> are zero-indexed too, so the first text column is '0', and the tenth
>>> text column is '9', etc... Let me know how it goes.- Zitierten Text ausblenden -
>
>>> - Zitierten Text anzeigen -
>
>> Hi Conor,
>
>> Thank you for the Code and all the explanations.I still don't get a
>> few points.
>> What is actually the meaning of "16" in the following statement:st =
>> [0,9,findgen(1000)*8 + 16]?
>>   is it the number of blanks in one of the line in the file above? and
>> what about

>> "+5" and 1002 in len = fltarr(1002) + 5?(is maybe 5 for the length of
>> the langest cha-
>> racter in a line and 1002 instead of 1000 because of the two first
>> columns which don't follow
>> the same pattern as the rest columns?).
>> Thank you for your attention
>> C.
>
> Sorry, I should have been more clear. So the goal is to make two row
> arrays, each with a number of elements equal to the number of columns
> in your file. So, for starters in the second line I used fltarr(1002)
> simply because the first array has 1002 elements. Essentially, the
> above example is for a file with 1002 columns.
>
> The second array (len) needs to have the length for every single
> column in the text file. fltarr(1002) + 5 makes a row array with 1002
> entries, each with the value "5". So, in this example the program
> would be expecting a maximum of 1002 columns in every line, and each
> section of data will be at most 5 characters long (if some data
> columns are slightly shorter than 5 characters it will be okay, as
> long as it only grabs spaces and doesn't start grabbing data from
> another column).
>
> The first array, st, is intended to be an array with an element for
> every column in the data file, specifying where each column of data
> starts. In the example you gave, data columns start at the points:
>
> [0,9,16,24,32,etc...]
>
> The latter, repeating sequence is basically findgen(n)*8 However, the
> sequence starts at 16, not at 0. findgen(n)*8 starts at zero, so to
> make it start at 16 I add 16 to every entry, and then add the first
> two columns on before it [0,9,findgen(1000)*8 + 16] Make sense?
> You'll probably have to do something similar for your data file.
> Assuming the example you gave is directly from your data file, and the
> layout doesn't change in later columns, then you would do:
>
> st = [0,9,findgen(1018)*8 + 16]
> len = fltarr(1020) + 5
>
> Just to be clear: you use findgen(1018) instead of findgen(1020)
> because you've already specified the first two columns, so you only
> have to generate the last 1018 columns with the findgen().- Zitierten Text ausblenden -
>
> - Zitierten Text anzeigen -

Hi Conor,

Hier ist how the whole code(I also read the header)looks like:

```
function parse_bigfile,filename

  file=strupcase(filename)

 ;Header definition
  header=strarr(5)

 ;Determine the number of rows in the file
  rows=file_lines(file)
 ; print,rows

 ;open the file and read the five line header
  openr,unit,file,/get_lun
  readf,unit,header

; Find the number of columns in the file
   cols=fix(strmid(header(3),14,4))
   print,cols

; Number of rows of the data
  rows_data=rows-n_elements(header)
;  print,rows_data

 st = [0,406,findgen(cols-2)*6+412]
 len = fltarr(cols)+5
 num = n_elements(len)

line = ''
data = intarr(num)

l = 0
while not( eof(unit) ) do begin

 ; read in the line and see how long it is
 readf,unit,line
 data = intarr(num)
 length = strlen(line)

 for i=0,num-1 do begin
  ; if we've moved past the end of the line, we are done with this
line
  if st[i] gt length-1 or length eq 0 then break

  ; read and process the current element
  data[i] = float( strmid( line, st[i], len[i] ) )
 endfor
```

```
 ; if this is the first line, create our data result.  Otherwise, just
append the new data
 if l eq 0 then result = data else result = [[result],[data]]

 ; increment our line counter
 ++l
endwhile

close,unit
free_lun,unit

return,result

end
```

I can't managed to read the file with or without header.I'm always getting the
following error message:
Type conversion error:Unable to convert given STRING to float.It's always crashing
at the statement:data[i] = float(strmid(line, st[i], len[i]))

Thank you for your attention
C.
