Subject: Re: A (too?) simple question about importing data
Posted by Craig Markwardt on Thu, 22 Jun 2000 07:00:00 GMT
View Forum Message <> Reply to Message

q4668057@bonsai.fernuni-hagen.de (Michael Spranger) writes:
> Hi,
> another beginner's question, this time about reading data:
> I want to read data from ASCII files into a structure. The data look
> as follows:
>
> YYYY MM DD HH II SSSSS PPPPPP LLLLLLL KKK RRR
> 0330 00 00 00 00 00000 50.60   03.40 000 0.0 USGS_EU_Catalogue
>
> the structure, type, and length of variables are always the same, only
> the the order might change and some data might be missing. The last
> row (without header) contains comments only.

It's a beginner's *and* advanced user's question. My suggestion is to try TRANSREAD available from my web page. It attempts to make it easy to read lots of data from a file. [To get the formatting right I suggest using the /DEBUG option.]

I made a file called test.dat with the following lines:

```
YYYY MM DD HH II SSSSS PPPPPP LLLLLLL KKK RRR
0330 00 00 00 00 00000  50.60   03.40 000 0.0 USGS_EU_Catalogue
0340 00 00 00 00 00000 124.56   03.40 000 0.0 Test line 1
0350 00 00 00 00 00000 789.01   03.40 000 0.0 Test line 2
```

And then executed the following commands:

```
IDL> yyyy = 0L & mm = 0L & dd = 0L & hh = 0L & ii = 0L & sssss = 0L
IDL> pppppp = 0D & lllllll = 0D & kkk = 0L & rrr = 0D & ccc = ''
IDL> transread, unit, yyyy, mm, dd, hh, ii, sssss, pppppp, lllllll, kkk, $
    rrr, ccc, format='(I5,I3,I3,I3,I3,I6,D7,D8,I4,D4,A0)', file='test.dat'
IDL> print, yyyy
       330       340       350
```

The first two lines establish the types of each variable -- I used the column headers you provided. The third line is the actual invocation of TRANSREAD. The format keyword is vital, and may take some experimentation. Note that lines that don't match the format are skipped automatically, you can define comment characters, and you can specify start/stop "cues" to enable/disable parsing.

Craig
http://cow.physics.wisc.edu/~craigm/idl/idl.html

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: A (too?) simple question about importing data
Posted by promashkin on Thu, 22 Jun 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Hi Michael,
If the data bearing strings are well-defined (e.g., data or filling with
"bad" number are present always), then the following would work:

```
; create a file with dummy data first...
temp = '0330 00 00 00 00 00000  50.60   03.40 000 0.0 USGS_EU_Catalogue'
; make 100 rows in that file
temp = replicate(temp, 100)
openw, unit, 'temp_junk.txt', /get_lun
printf, unit, temp
free_lun, unit
; now we have a file to try to read.
; open the file for reading
openr, unit, 'temp_junk.txt', /get_lun
; Create STR_FORM that reflects format of data in one file row
str_form = {data:fltarr(10), note:''}
; create array of STR_FORMs big enough to read the whole file at once.
; lets pretend we don't know file length in advance.
data_array = replicate(str_form, 2000)
; in this case it is way too big. Not to worry.
readf, unit, data_array
;% READF: End of file encountered. Unit: 100
;       File: IDE data:idl:ukmo:temp_junk.txt
;% Execution halted at:  $MAIN$
; Sure enough, reading failed. But we know file size now.
; The number of fields (10 values and a string) is 11, so we do:
print, (fstat(unit)).transfer_count / 11
;        100
; this means we had 100 rows in the file. Resize the array:
data_array = replicate(str_form, 100)
; start over in the file:
point_lun, unit, 0
; read the array:
readf, unit, data_array
print, data_array[2]
;{     330.000      0.00000      0.00000      0.00000      0.00000
;      0.00000      50.6000      3.40000      0.00000      0.00000
```

; USGS_EU_Catalogue}

I discovered (for myself - the Pros knew that all along, I'd think :-)
that reading past the end of file and then resizing the read buffer is a
lot faster than reading accurately line by line inside a WHILE NOT EOF
loop. IDL can read a 100x100000 FLTARR directly a thousand times faster
than going through a 100000 line loop, reading a 1000 point vector at a time.

Will this work?
Cheers,
Pavel

Michael Spranger wrote:
>
> Hi,
> another beginner's question, this time about reading data:
> I want to read data from ASCII files into a structure. The data look
> as follows:
>
> YYYY MM DD HH II SSSSS PPPPPP LLLLLLL KKK RRR
> 0330 00 00 00 00 00000 50.60   03.40 000 0.0 USGS_EU_Catalogue
>
> the structure, type, and length of variables are always the same, only
> the the order might change and some data might be missing. The last
> row (without header) contains comments only.
>
> Sounds easy, is (probably) easy - but (still) too difficult for me.
>
> Thanks for any help/ hints in advance,
> Michael

Subject: Re: A (too?) simple question about importing data
Posted by promashkin on Fri, 23 Jun 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Michael,
It seems to me that you can easily accomplish what you want by reading
the header row of your file and defining the STR_FORM and DATA_ARR (same
context as in my earlier example) from that:

```
; if the first line of the file is header, then
readf, unit, header_string
; turn it into string array
header_string = strsplit(header_string, /extract)
; Create STR_FORM differently, using TEST (compile it first):
function test, x
```

```
str_form = create_struct(x[0], 0.0)
for i=1, n_elements(x)-1 do begin
 str_form = create_struct(str_form, x[i], 0.0)
endfor
str_form = create_struct(str_form, 'NOTE', '')
return, str_form
end
; now, STR_FORM has fields with names from header string.
; since USGS... string is not in header, we add it separately.
DATA_ARR = replicate(test(header_string), 100)
; now we can read the file. Note that we can keep reading,
; because the cursor is at start of data section already.
readf, unit, data_arr
free_lun, unit
```

This provides you with array of structures, with fields named according
to your header line.
The only thing is, I see no way how you could make your code "guess"
whether a column is numerical or a string, unless you go through a
painful way of reading in a string (or string array) and doing STRSPLIT,
at least once for each file. This is the last resort I would use, and I
had sometimes when I got desperate with very wierd, inconsistent files.
If you have small number of column headers and they always are the same
(lets say, LLLLLLL is always FLOAT, PPPPPP is FLOAT etc.) you can easily
write a lookup table with CASE statement and add it to the TEST function
to define the type of fields in STR_FORM correctly. It will not slow you
down much because you define STR_FORM only once. Of course, if they all
are always numerical, then everything will work as it is.
The good thing about this approach is that you can work with DATA_ARR_1
that has 10 columns, or DATA_ARR_2 that has only 5, without much
difference, like follows, if the columns you address are present in both files:

```
plot, data_arr_1.yyyy, data_arr_1.PPPPPP
plot, data_arr_2.yyyy, data_arr_2.PPPPPP
```

Hope this helps.
Cheers,
Pavel

Michael Spranger wrote:
>
> Thanks Craig and Pavel,
>
> your both answers solved the direct problem perfectly (and saved me a
> lot of time) - I originally intended to find a more general solution,
> as I receive these datafiles sometimes with resorted columns. I wanted
> to read the array automatically depending on the position of the
> corresponding characters in the header row.

> (it might also be 'PPPPPP LLLLLL YYYY ...'
> Probably it is far easier and faster to reformat the data beforehand
> than spending hours on this problem.
>
> Michael

Subject: Re: A (too?) simple question about importing data
Posted by q4668057 on Fri, 23 Jun 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Thanks Craig and Pavel,

your both answers solved the direct problem perfectly (and saved me a lot of time) - I originally intended to find a more general solution, as I receive these datafiles sometimes with resorted columns. I wanted to read the array automatically depending on the position of the corresponding characters in the header row.
(it might also be 'PPPPPP LLLLLL YYYY ...'
Probably it is far easier and faster to reformat the data beforehand than spending hours on this problem.

Michael
