Subject: Counting header lines in a file
Posted by thtran296 on Fri, 21 Jul 2017 20:59:06 GMT
View Forum Message <> Reply to Message

Hello guys,

I have a .dat file that looks like this:

Date: May 5, 2016
Name: a person's name goes here
Experiment with temperature blabla

Day. Temperature
1 56
2 62
3 63
4 95

_____
Anyway, you get the idea.
So I'm trying to read in just the numeric part of the data (the 2 cols of numbers), and ignore the headers.
Here's what I have so far:

```
lineofheaders = 5
file = 'tryhard.dat'
rows = file_lines(file)                ;count rows of entire file
openr, lun, file, /get_pun
header = strarr(lineofheaders). ;pre-allocate to read the header
readf, lun, header
point_lun, -lun, currentlocation
line =""
readf, lun, line
cols = n_elements(float(strsplit(line, /regex)))
data = fltarr(cols,rows - lineofheaders)
point_lun, lun, currentlocation
readf, lun, data
free_lun, lun
```

The above code did work, of course. However, my problem is that I always have to know in advance the number of lines that the header takes up. For example, in this file the header takes up 5 lines, so I will start reading data from line 6 till end of file.

Is there any way to still do the above, without knowing in advance how many lines the header takes up?

Thank you so much.
Thomas

Subject: Re: Counting header lines in a file
Posted by Nikola on Sat, 22 Jul 2017 10:26:33 GMT
View Forum Message <> Reply to Message

On Friday, July 21, 2017 at 9:59:09 PM UTC+1, thtr...@gmail.com wrote:
> Hello guys,
>
> I have a .dat file that looks like this:
>
> Date: May 5, 2016
> Name: a person's name goes here
> Experiment with temperature blabla
>
> Day. Temperature
> 1 56
> 2 62
> 3 63
> 4 95
>
> _____
> Anyway, you get the idea.
> So I'm trying to read in just the numeric part of the data (the 2 cols of numbers), and ignore the headers.
> Here's what I have so far:
>
> lineofheaders = 5
> file = 'tryhard.dat'
> rows = file_lines(file) ;count rows of entire file
> openr, lun, file, /get_pun
> header = strarr(lineofheaders). ;pre-allocate to read the header
> readf, lun, header
> point_lun, -lun, currentlocation
> line =""
> readf, lun, line
> cols = n_elements(float(strsplit(line, /regex)))
> data = fltarr(cols,rows - lineofheaders)
> point_lun, lun, currentlocation
> readf, lun, data
> free_lun, lun
>
> The above code did work, of course. However, my problem is that I always have to know in advance the number of lines that the header takes up. For example, in this file the header takes up 5 lines, so I will start reading data from line 6 till end of file.
>
> Is there any way to still do the above, without knowing in advance how many lines the header takes up?
>
> Thank you so much.
> Thomas

Once you read a line, you can parse it using the string functions. For example, strmid(line, 0, 1) returns the first character of the string. Then you can test if it is a letter or a number so that you can decide if it belongs to the header or to the data.

Subject: Re: Counting header lines in a file
Posted by Markus Schmassmann on Mon, 24 Jul 2017 15:30:04 GMT
View Forum Message <> Reply to Message

On 07/24/2017 03:56 PM, thtran296@gmail.com wrote:
>> Once you read a line, you can parse it using the string functions. For example, strmid(line, 0, 1) returns the first character of the string. Then you can test if it is a letter or a number so that you can decide if it belongs to the header or to the data.
>
> The function strmid() returns a string to me, even if the input is a number.
> For example,
> a = 123456
> result = strmid(strtrim(a,2),1,4) & print, result & help, result
> IDL print:
> 2345
> STRING = '2345'
>
> For this reason, when I use the ISA() function to test if it is a string or number, it would return string to everything. So how can I test if it is a letter or a number? Is there another function besides ISA() ?
I would recommend using the STRMATCH function to test whether your line
contains numbers. Example:

```
tab=string(9b)
regex1='^[\ '+tab+']*[0-9]*[\ '+tab+']*[0-9]*[\ '+tab+']*$'

openr, lun, file, /get_lun
header=!null
repeat begin
    point_lun, -lun, currentlocation
    readf, lun, line
    header=[header,line]
endrep until strmatch(line,regex1)

header=header[0:-2]
point_lun, lun, currentlocation

data = fltarr(cols,rows - n_elements(header))
.....
```

Often used, but very bad programming style is to abuse the error

handling system.

```
function isa_number, string, number=number
err_no=0
catch, err_no
if err_no ne 0 then begin
    catch,/cancel
    message, /reset
    return, 0b
endif
number=0
reads, string, number
return, 1b
end
```

Both approaches need to be refined before use, (e.g. float instead of integers), but the idea should be clear.

Good luck, Markus

Subject: Re: Counting header lines in a file
Posted by Matthew Argall on Wed, 26 Jul 2017 12:59:20 GMT
View Forum Message <> Reply to Message

Here is how I solved this problem.
https://github.com/argallmr/IDLlib/blob/master/file_utils/mr file_read_ascii.pro

At the top, there is a helper function called "MrFile_Read_Ascii_Header". It reads each line of the file 1-by-1 until it has read five consecutive lines with the same number of columns. All lines with mismatched number of columns are considered the header. Additionally, the first line of data is parsed to determine its formatting. The output can then be passed to Read_Ascii in the form of a template.

The main program, "MrFile_Read_Ascii", is a wrapper for the Ascii_Template and Read_Ascii procedures and is based off of a program from Mike Galloy.

Hope this helps

Subject: Re: Counting header lines in a file
Posted by thtran296 on Wed, 26 Jul 2017 14:01:06 GMT
View Forum Message <> Reply to Message

On Wednesday, July 26, 2017 at 8:59:23 AM UTC-4, Matthew Argall wrote:
> Here is how I solved this problem.

> https://github.com/argallmr/IDLlib/blob/master/file_utils/mr file_read_ascii.pro
>
> At the top, there is a helper function called "MrFile_Read_Ascii_Header". It reads each line of the file 1-by-1 until it has read five consecutive lines with the same number of columns. All lines with mismatched number of columns are considered the header. Additionally, the first line of data is parsed to determine its formatting. The output can then be passed to Read_Ascii in the form of a template.
>
> The main program, "MrFile_Read_Ascii", is a wrapper for the Ascii_Template and Read_Ascii procedures and is based off of a program from Mike Galloy.
>
> Hope this helps

I appreciate your help. That really is another great way to do this!
I have figured it out a few days ago but forgot to let you all know.
Basically I used the "strmatch" function to compare the first character of each line with the number 0 to 9. If it doesn't match, then that means the line must have started with a letter, and therefore is a header.
It worked wonderful!
The only caution was that I had to strtrim() each line to make remove the space in front of each line so that strmatch won't compare a space to a number.

Thank you all.
