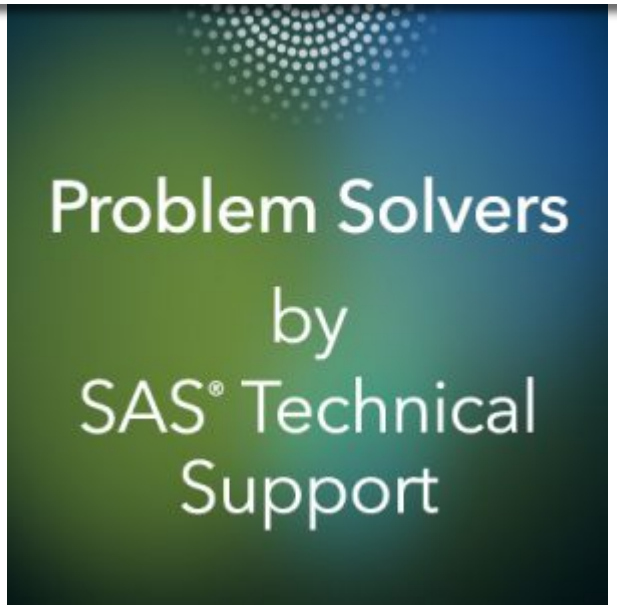
Turning external files into SAS® data sets: common problems and their solutions

By Amber Elam on SAS Users | February 18, 2021

Topics | Programming Tips

Reading an external file to create a SAS data set can either be easy or challenging depending on which tool outside of SAS created the external file. This post covers the two most common ways to read an external file and provides solutions for seven scenarios that you might encounter when reading external data.

The two most common ways to read an external file are using the IMPORT procedure or using a DATA step with INFILE and INPUT statements. For more information about reading external files using PROC IMPORT, see my earlier post, Tips for using the IMPORT procedure to read files that contain delimiters.



DATA step with INFILE and INPUT statements

Here is example code to read a comma-delimited file using the DATA step with INFILE and INPUT statements:

```
data one;
infile 'c:\cars.csv' dlm=',' dsd firstobs=2 truncover;
input make : $10. model : $20. type $ origin $ releasedt : mmddyy10. drivetrain $ msrp : comma10.;
run;
```

Here are details about the options in the above INFILE statement:

- The DLM= option indicates that the file is comma delimited.
- The DSD option specifies that multiple commas in a row should be treated as a missing value.
- The FIRSTOBS=2 option designates that the first row of data is the header record and should be skipped.
- The TRUNCOVER option indicates that the program should not flow over to the next line to look for missing values that are not found on the current line being read.

Note: This code did not include the LRECL= option because the default record length in SAS® 9.4 is 32,767. My external file did not

A few caveats and clarifications

In the example INPUT statement, I used modified list input and list input based on the value of the variable that I am reading in. If the value is:

- Numeric, I do not have to put anything after the variable name.
- A character eight characters or less, then I use a $ after the variable name to indicate that it is a character variable.
- A character greater than eight characters, I use the colon (:) followed by $w.informat to indicate the maximum width of the value. The colon indicates that the value should be read from the next non-blank value until either the delimiter or the width specified by the informat is reached, whichever comes first.

For the variable RELEASEDT, the value is a date in the form of *mm/dd/yyyy*, so I use the colon and the MMDDYY10. informat to read in the value.

For the MSRP variable, the value is in the form of $35,345, so I use the colon and the COMMAw. informat to read it in. In both cases, the value can vary in length, so I used the colon and the maximum width for the w. portion of the informat.

Troubleshooting scenarios

Now, let's talk about some troubleshooting that you might need to do when reading an external file using the DATA step with INFILE and INPUT statements.

Scenario 1: Specifying a tab-delimited file

Solution: Indicating that a file is tab-delimited is different from specifying files that are comma- or pipe- (|) delimited. You cannot type a tab when using the DLM= option, so you must use the hexadecimal representation of the tab. If you are using SAS on Microsoft Windows, UNIX, or a Mac, use the following syntax when reading a tab-delimited file:

```
Infile 'host-specific-path' dlm='09'x dsd <other options>;
```

If you are using SAS on MVS or z/OS, use the following syntax when reading a tab-delimited file:

```
Infile 'host-specific-path' dlm='05'x dsd <other options>;
```

Scenario 2: Reading in all the records

Issue: The external file that I am reading contains 1,252 records. When I read in the external file using the DATA step with INFILE and INPUT statements, SAS creates a data set with 531 observations. Why didn't SAS read all the records in the external file?

Solution: This situation is unique to the Windows operating system. Windows inserts three special characters to signal the end of the file. The characters are called an end-of-file (EOF) marker and the hexadecimal representation for the EOF marker is **'1A'x**. Sometimes an EOF marker can appear in the middle of your data instead of at the end of the external file. When SAS encounters the hexadecimal **1A**, SAS considers this the end of the file and stops reading the data. You can use the IGNOREDOSEOF option on the INFILE statement or FILENAME statement to indicate that those premature EOF markers in the middle of your data can be ignored. Here is example code:

```
Infile 'c:\abc.txt' ignoredoseof <other options>;
```

Scenario 3: Reading in files that contain multiple records

```
NOTE: 1 record was read from the infile 'c:\temp\classfixed.txt'.
      The minimum record length was 380.
      The maximum record length was 380.
NOTE: The data set WORK.ONE has 1 observations and 5 variables.
```

How can I read in all my records in the external file?

Solution: From the information in the SAS log, you have fixed records within one long stream of data. To address the issue, you need to use two options in the INFILE statement: RECFM=F and LRECL=*xxx* where *xxx* is the exact record length. When you specify these two options, SAS reads the first *xxx* characters based on the LRECL= value for the first observation. The next *xxx* characters will be the next observation and so on. In your INPUT statement, you need to list each variable and the number of columns (width) for each variable to ensure that it matches the length specified for the LRECL= value.

Here is an example:

```
data one;
infile ' host-specific-path' recfm=f lrecl=25;
input name $10. age 2. class $10. room 3.;
run;
```

If you add up the widths specified after the NAME, AGE, CLASS, and ROOM variables, it equals 25, which is the value used with the LRECL= option.

Scenario 4: Preventing truncated values

Issue: I'm reading an external file with a long record length, which is causing some of my data values to be truncated. What do the following messages in the SAS log mean?

Solution: These notes indicate that some of the records in your external file exceed the default LRECL= value of **32,767**. As a result, any data values past column 32,767 are not read in.

Add the LRECL= option to the INFILE statement and use a value greater than **32,767**. Here is example code:

```
Infile 'host-specific-path' lrecl=50000 <other options>;
```

Beginning in SAS 9.4, the default LRECL= value is **32,767**. The LRECL= value can range from **1** to **1,073,741,823** (1 gigabyte) on Windows and UNIX.

Scenario 5: Preventing flowover

Issue: What does the following note in the SAS log mean?

```
NOTE: SAS went to a new line when INPUT statement reached past the end of a line.
```

Solution: This note indicates that SAS is flowing over to the next line to look for values not found on the current line being read. This is the default behavior when SAS is not able to find the values being read on the current record. To prevent a flowover, use the TRUNCOVER option in the INFILE statement, as shown below:

```
Infile 'host-specific-path' truncover <other options>;
```

Scenario 6: Avoiding missing data

Issue: The last variable in my external file is numeric. When SAS reads in the variable, it is missing for every observation. Here is

```
NOTE: Invalid data for d in line 1 7-8.
RULE:     ----+----1----+----2----+----3
1   CHAR  1,2,3,4. 8
    ZONE  32323230
    NUMR  1C2C3C4D
a=1 b=2 c=3 d=. _ERROR_=1 _N_=1
```

How do I read in the data correctly?

Solution: The most common cause for this situation is reading a Windows file on a UNIX operating system. A file that is created on Windows operating system has a Carriage Return/Line Feed (CR/LF) at the end of each record. A file that is created on UNIX operating system has a Line Feed (LF) at the end of each record. Therefore, a file that is created on Windows and transferred over to UNIX still has a CR/LF at the end of each record.

The Carriage Return (CR) becomes part of the value of the variable. Since the variable was being read in as numeric, you receive the above "Invalid data" message in the SAS log. In the SAS log above the message, the ruler, and the CHAR/ZONE/NUMR group show a period at the end of your number. Below the period in the ZONE line is a 0; below that in the NUMR line is a D. These are elements of **'0D'x** (the hexadecimal representation of a CR), but SAS is reading the hexadecimal value as actual data.

To address the issue, add the TERMSTR= option to the INFILE statement. Specifying TERMSTR=CRLF, as shown below, indicates that the file has the Windows End-of-Record (EOR) markers instead of UNIX EOR markers:

```
Infile 'host-specific-path' termstr=crlf <other options>;
```

The TERMSTR= option is valid in the INFILE statement or the FILENAME statement.

Beginning in SAS 9.4, TERMSTR= is set to CRLF by default when running on UNIX. For more information, see SAS Note 66323, "A UNIX operating system can now read a Windows file without specifying the TERMSTR= option."

Issue: The comma-delimited file that I read in shows more observations in the SAS data set than the number of records in the external file. Also, some of my variable values in a new observation have values for variables from the previous observation.

Solution: You could have a quoted string for one of your variable values. The quoted string contains the EOR marker, such as a CR/LF or a LF, in the middle of the string as part of a data value. The hexadecimal representation for CR/LF is **'0D0A'x**. The hexadecimal representation for LF is **'0A'x**. SAS considers CR/LF or LF to represent the end of a record. Therefore, you end up with more observations and/or many "Invalid data" messages in the SAS log.

For example, sometimes a long comment field stops in the middle and continues to the next record in a CSV file. This problem occurs mostly in files that are created from Microsoft Excel. In Excel, columns contain soft returns to help with formatting. When you save the worksheet as a CSV file, the soft returns appear to SAS as EOR markers. This, in turn, causes the records to be split incorrectly.

If your data value that contains the EOR marker is enclosed in quotation marks, you can use the sample code in SAS Note 26065 to remove the erroneous CR/LF or LF that is part of the quoted value. After you use the code in the SAS Note to pre-process the external file and remove the erroneous CR/LF or LF, then you should be able to read the external file and return the correct number of observations.

If your data value that contains the EOR marker is NOT enclosed in quotation marks, you need to use a tool outside of SAS to pre-process the external file to remove the erroneous CR/LF or LF characters. In this situation, SAS cannot determine what is a true EOR marker versus an erroneous EOR marker.

The best way to look at the CSV file to determine whether your data values that contain the EOR marker are quoted is to use a tool like Notepad or TextPad. Excel does not show the data in a format that displays the erroneous EOR marker.

Final thoughts

If you encounter an issue when reading in an external file, examine the SAS log for any notes, errors, or warnings. Hopefully, one of the scenarios above matches what you are encountering. If not, Technical Support is here to help—just give us a call or send us an email! If you send an email, include your site number, SAS Release including maintenance level, operating system, a brief description of the problem, your SAS log as an attachment from running the code in a new SAS session, and a sample of the external file (if possible) as an attachment.

CONTACT TECH SUPPORT | CHOOSE A CHANNEL

Tags

SAS Programmers | reading text files | Problem Solvers

Share

ABOUT AUTHOR

Amber Elam

Technical Support Analyst

Amber Elam is a Technical Support Analyst in the Foundation SAS group in Technical Support. She has worked in

21 COMMENTS

Tamara Barker on April 23, 2021 9:11 pm

There are some great tips here! I will be sharing a link to this post with my SAS users. Thank you!

Reply >

Leonid Batkhan on February 19, 2021 9:40 am

Great post, Amber! Very useful information.

Reply >

Lisa on February 18, 2021 7:04 pm

Excellent blog, Amber!

Reply >

Tam on September 4, 2019 4:59 pm

What is the email we can ask questions. I have a problem with data reading in sas.

Reply >

Shelley Sessoms on September 5, 2019 7:58 am

You have several ways to contact Tech Support: https://support.sas.com/en/technical-support.html

Best of luck to you.

Reply >

Thank you so much. I spent good 5-6 hour trying to find the fix, and the solution was one simple option: IGNOREDOSEOF

Reply >

John Wang on April 4, 2018 10:48 pm

Hi Charley:
I got a question for you. The txt file is as following. My program is next to it. The last two columns of SEXLD and BYRLD of my SAS data contain the read through values, which should be Null or Missing (the other columns are perfectly fine). Can you help me out on this?

Thanks,

John Wang

CEMA401A2201||UN_TR_RES_MED_LD|RESCUE TREATMENT|ACETAMINOPHEN/PARACETAMOL|1.0000|TABLET|ORAL|Post-Herpetic Neuralgia|20170828||Y|Y|LOADED||
CEMA401A2201|1021001|UN_TR_RES_MED_LD|RESCUE TREATMENT|ACETAMINOPHEN/PARACETAMOL|3.0000|mg|ORAL|Post-Herpetic Neuralgia|20171024||Y|Y|LOADED||
CEMA401A2201|1021001|UN_TR_RES_MED_LD|RESCUE TREATMENT|ACETAMINOPHEN/PARACETAMOL|1000.0000|mg|ORAL|Post-Herpetic Neuralgia|20171030||Y|Y|LOADED||
CEMA401A2201|1021001|UN_TR_RES_MED_LD|RESCUE TREATMENT|ACETAMINOPHEN/PARACETAMOL|1000.0000|mg|ORAL|Post-Herpetic Neuralgia|20171106||Y|Y|LOADED||

```
data CM_pre;
infile "&CM_pre." dsd dlm='|' LRECL=400 truncover;
input STUDY :$15. PATIENT :$10. CPEVENT :$20. CMCAT :$80. TRTCM :$70. CMDOSE :8. CMDOSU :$80. CMROUTE
```

```
CMSTDAT :$8. CMSTTIM :$6. OCCURCM :$2. CMPRESP :$2. DATLDFL :$10. SEXLD :$2 BYRLD :4.;
run;
```

Reply ›

Charley on February 26, 2018 12:13 pm

Hello Nick,
One of the tools that I've used in the past is to modify the input buffer to alter or remove unwanted characters. It looks like this:

```
data new;
infile whatever dlm="~" dsd truncover;
input @; /* this pulls a record from the file into the input buffer and holds it*/
_infile_=compress(_infile_,"'"); /* this removes the single quotes from the record in the buffer*/
input @1 a b c d ... ; /* and this parses the data into the variables */
run;
```

Reply ›

Nick on February 21, 2018 4:14 pm

Hi, I have question regarding delimiters. How to define delimiter which consist quotation marks?
Data looks like this: 1234'~'2345'~'3455'~'456788
When I define this delimiter by using dlmstr (dlmstr='~') command, I receive in the result : 1234' '2345' '3455' '456788
I need results without quotation marks

Reply ›

Mathew Griffiths on March 28, 2017 4:46 am

Hi, I have a problem reading in a raw data file and I have been searching the internet for a while now on how to fix it.

I have a comma delimited file, and am using the DSD option. However, some of my variables that should be enclosed in quotes, are not. Therefore DSD option does not recognise that this should all be in one variable.

Any help on how to read this in?
Thanks
Mathew

Reply ›

Charley on March 28, 2017 12:04 pm

Hello Mathew,
Open a track and attach the file or a reasonable sample of it. The file may yet be readable. I'm going to need to look at it and see if I can find a pattern that will allow SAS to read it.

Reply ›

James Rees on February 9, 2017 10:34 am

hi I am trying to read in the following file and it fails to read in the last field customer_type every time. It was originally saved on an iMac as a csv file

Customer_ID,Country,First_Name,Last_Name,Birth_Date,Customer_Type
4,US,James,Kvarniq,27/06/1974,”Orion Club members low activity”
12,US,David,Black,12/04/1969,”Orion Club members medium activity”
13,DE,Markus,Sepke,21/07/1988,”Orion Club Gold members low activity”
16,DE,Ulrich,Heyde,16/01/1939,”Internet/Catalog Customers”

Reply ›

Charley Mullin on February 9, 2017 1:07 pm

I pasted the 6 lines in to a .csv file using a plain text editor then used PROC IMPORT to read the data. The data set was created correctly except for the double-quotes being read as part of the text.
On closer inspection of the file I discovered that the double-quotes in the file are not what SAS is expecting. The double-quotes in the file have a hex value of '94'x. The double-quotes that SAS is expecting have a value of '22'x. So the double-quotes are treated as data and may or may not be displayed correctly depending on the font you are using.
If you are running SAS 9.2 or above, you can record the existing characters into what SAS is expecting and avoid further issues.
Here is the code that I used to edit the file in place and PROC IMPORT to create the data set:

```
data _null_;
infile 'c:\_today\james.csv' lrecl=32767 truncover sharebuffers;
file 'c:\_today\james.csv' lrecl=32767 ;
input ;
_infile_=translate(_infile_,'','94'x);
put _infile_;
run;

proc import datafile='c:\_today\james.csv'
dbms=csv out=james replace;
run;
```

Reply ›

Andrea Magatti on February 7, 2017 6:36 am

Simple question, i've got a file where one text field, usually has no double quotes as text delimiter, but on some rows that text field has a trailing double quote (not closed)
How to deal with that?

Reply ›

I'd like to see a sample of the file. It might be easiest to open a track and attach a sample. I've seen a few files where delimiters and double-quotes needed to be counted and handled case by case if there are an odd number of double-quotes.

Reply >

Rakesh on November 8, 2016 5:51 am

Hi Chris,

I am trying to pipe in the test in infile and reading values from a server based table, the problem is SAS is reading alternate values rather than all values from piping text.

any help on this?

Reply >

Indra on October 22, 2016 2:31 am

very Good Information! Thanks Charley

Reply >

Charley Mullin on April 20, 2015 4:32 pm

Chris, Robert. Thank you very much. I appreciate the feedback.

Reply >

Robert Allison on April 20, 2015 9:31 am

Good stuff!

Chris Hemedinger on April 17, 2015 10:41 am

Great post Charlie! One more thing that I've (re)learned recently -- the INFILE statement can process wildcard specifications so you can read in multiple files with the same structure in a single step. The FILENAME= option can be used to squirrel away the name of the file you've processed.

```
infile "c:\project\*.csv" filename=infileName;
```

Reply >

Justin on May 9, 2017 10:56 am

If you use the above approach, what do you do in the initial DATA step line where you typically name the SAS data set?

Reply >

LEAVE A REPLY

Your Comment

Your Name

Your Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment