

# **Data Entry and Data Transfer Errors What They Are, How to Prevent Them and How to Correct Them**

Iris McPherson, Oklahoma State University

## **Abstract**

The different methods for entering, uploading, downloading, and ftping the data to different platforms create a whole new set of problems for the SAS programmer. The data is entered in one form, but by the time it finally arrives at the machine where it will be used by SAS the characteristics of the file may have changed or the data may have been altered. Basing your INPUT statement on what the data looked like at its source can be disastrous. The first step in avoiding such problems is being aware that the potential for problems exists. Next the data and the file characteristics should be carefully examined. Once any problems are identified, use SAS options and functions and host commands available to get the data in a usable form again. The final step in this process, and one that you should always remember, even when no problem appears to exist, is to check the data after it has been read by the SAS program to make sure it has been read correctly.

## **Introduction**

Advances in computing now give us the capability of entering data using a wide variety of software packages, most of which have a feature that allows us to save the data in text format, so it can be read by other software packages. We can also move data between platforms in numerous ways. On the surface this seems to be a fairly successful operation. Unfortunately, that may in fact not be the case. The move itself may have been made successfully, but when the file is accessed by a SAS program problems may crop up. The worse case scenario is one in which there are problems reading the file, but they are not noticed by the programmer or user, because the program appears to run. Another common situation is where the programmer or user realizes something is wrong, but since the data often looks correct when viewed, they are at a loss to know what is causing the problem. In this paper we will look at specific problems caused by data entry and data transfer (PC-based systems to MVS, CMS and OpenVMS). Visual data checking and perusal of the SAS log will be used in detecting problems. I will discuss options in data transfer software, options on the SAS INFILE statement and mainframe commands which you can use to prevent or correct problems. The PRINT procedure is used to determine if all the preventive and/or corrective measures have been effective.

## **Know the Potential Problems**

The best defense against problems in moving data between applications and platforms is to be aware that the possibility for such problems exists. It is always a good idea to look at the data after a move to see if there are any obvious problems, but don't trust your eyes, even though everything looks correct. Some data transfer issues are:

- When a file is transferred, the default file characteristics of the transfer software package are used unless they are overridden or the file on the host machine is defined before the move is made. These default file characteristics may not be compatible with the type of INPUT statement you are using in your program.

- When transferring data between platforms there may be a default setting for the use of a carriage return and line feed at the end of each record. Unless the default creates a carriage return and line feed, there will be a problem.
- When moving files with some software an extra line may be added at the end of the file. This is sometimes a blank line and other times may be a line of unprintable characters.

There have always been problems caused by data entry. With our current capabilities in data entry another group of data issues has been added. Some of the major issues are:

- When you enter your data using a package other than SAS you save the file in text format, so that it can be read easily by your SAS program. Some of these packages give you a choice of delimiter to use to separate the values. The default data delimiter in SAS is a blank.
- When a text file is created you need to know how missing values are indicated. A common method is to have two adjacent delimiters. SAS expects a blank unless list input is used then it expects a period.
- When there are character values in a text file they may be enclosed in quotes. The default in SAS is to include the quotes as part of the value.
- When a delimiter is included as part of a character value, it is treated as a data delimiter.
- When the variable names appear as the first record in the text file, this record will be character while the remainder of the records will more than likely be numeric or a mix of character and numeric. The default in SAS is for each observation to contain the same type of data.
- When current data entry methods are used, we are seldom restricted to using only 80 columns per logical record. For SAS on mainframe operating systems the default logical record length is the existing record length, but in SAS for PC operating systems the default length is 132.

### **What to Look for in the Data and the SAS Log**

It is always a good idea to look at your data before you use it, no matter how or where it was originally entered. Some errors will stand out, such as the following:

- Characters where there should only be numbers

```
01 1 15 17 152
02 1 22 12 116
03 2 27 15 133
04 F 21 13 145
05 2 19 20 16p
```

- Lines that wrap rather than each record starting on a new line (example of a file transferred without carriage return and line feed)

**Original file**

```
011 72 160
012 50 30
021 65 130
022 15 5
031 69 155
032 20 7
041 75 200
042 75 50
```

```
011 72 160""012 50 30""021 65 130""022 15 5""031 69 155""032 20
7""04 (a continuation of the first line)
1 75 200""042 75 50""
```

- Unprintable characters (in previous example "s represent unprintable characters on CMS, on MVS in SPF a message about unprintable characters would appear at the top of the file)
- Lines that are truncated (using the same original file as the previous example read by a program with the logical record length set at 10)

```
011 72 16
012 50 3
021 65 13
022 15
031 69 15
032 20
041 75 20
```

- Lines that are of varying lengths

```
011 72 160
012 50 30
021 65 130
022 15 5
031 69 155
032 20 7
041 75 200
042 75 50
```

- Unexpected blank lines

```
011 72 160
012 50 30
021 65 130
022 15 5
031 69 155

032 20 7
041 75 200
042 75 50
```

Even if your records are 72 characters or less, you should scroll the file to the left, to see if there is anything else on the record. This is especially true if you are using MVS, because of the numbered and unnumbered data sets.

Users often overlook the SAS log, or at best they only check the end of the log for error messages. Many of the indications of problems of the type we're discussing aren't errors, but rather show up as notes. The ERROR and NOTE statements at the end of each SAS data step are especially important in the type of checking we are doing.

The file characteristics SAS used in reading the file are given at the end of the data step. Many of the packages and protocols used to move data between computers have default file characteristics of record format of V (variable) or VB (variable blocked) and a logical record length of 255. A record format of V or VB means that the records do not have to be the same length. The length of each record is determined by the location of the last nonblank character encountered in the record. It is possible that all the records will be the same length, but even that doesn't guarantee there won't be any problems. From the file characteristics given in the SAS log you can see what the record format, logical record length, and minimum and maximum length records in the file are. There will be problems, if the minimum and the maximum are not the same. Even when the lengths are the same there may still be problems.

```
NOTE: The infile 'UCCXILM.RECFMVB.DATA' is:
      Dsname=UCCXILM.RECFMVB.DATA,
      Unit=3390,Volume=OSU013,Disp=SHR,Blksize=259,
      Lrecl=255,Recfm=VB
```

```
NOTE: 5 records were read from the infile 'UCCXILM.RECFMVB.DATA'.
      The minimum record length was 1.
      The maximum record length was 26.
```

Another indication that something is wrong with the logical record length or the INPUT statement is the following line in the SAS log:

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

This is due to the FLOWOVER option, the default on the INFILE statement. When this option is in effect and the INPUT statement reads past the end of the current record, then it continues reading data from the next record.

A message that often gets overlooked is the following NOTE:

```
NOTE: LOST CARD.
A=21 B=31 C=41 D=51 E=. _ERROR_=1 _N_=1
```

This will happen if the situation we just discussed occurs on the last observation. SAS tries to go to the next line to finish the INPUT statement, but there are no more lines, thus the message. Another common problem that causes this NOTE to appear is when you have observations that require more than one line of input, and the total number of lines read is not a multiple of the number of lines/observation. This can be due to extra lines that have been added during file transfer or data entry errors that produce too few or too many lines/observation. You should always look for the source of this NOTE.

```

data tryit;
infile cards missover;
input id 1-2 line 3 ht_in 5-7 wt_lbs 9-11
      #2 situps 5-7 pushups 9-11;
cards;
011 72 160
012 50 30
021 65 130
022 15 5
031 69 155
032
032 20 7
041 75 200
042 75 50
proc print;
run;

```

NOTE: LOST CARD.

```

RULE:-----1-----2-----3-----4-----5-----6-----
--7---
45  042 75 50

```

| OBS | ID | LINE | HT_IN | WT_LBS | SITUPS | PUSHUPS |
|-----|----|------|-------|--------|--------|---------|
| 1   | 1  | 1    | 72    | 160    | 50     | 30      |
| 2   | 2  | 1    | 65    | 130    | 15     | 5       |
| 3   | 3  | 1    | 69    | 155    | .      | .       |
| 4   | 3  | 2    | 20    | 7      | 75     | 200     |

The INVALIDDATA message is harder to ignore than some of the ones already mentioned, since it can increase the size of the SAS log substantially. This message lists the problem variables in an observation and the columns from which they are being read. It lists the line causing the problem, then lists the variables and the value SAS read for each one as shown here:

**NOTE: Invalid data for SCORE in line 4 6-8.**

```

RULE:-----1-----2-----3-----4-----5-----6-----
4  01M 15 72
ID=1 SEX=M AGE=1 SCORE=. _ERROR_=1 _N_=1

```

**NOTE: Invalid data for AGE in line 5 4-5.**

```

5  02 F1655
ID=2 SEX= AGE=. SCORE=655 _ERROR_=1 _N_=2

```

Even with the increase in the bulk of the output I'm amazed at how many people ignore the message. I would think they would at least want to get rid of all those extra lines of output. There are a number of the problems discussed earlier that can produce this message. When the first observation produced in a text file contains the variable names, the INVALIDDATA flag is set unless all the variables in the INPUT statement are defined as character. A file that has had unprintable characters added at some point in the process will also raise the flag.

In the SAS log at the end of each data step there are NOTES that tell you how many records were read from the input file and how many observations were written to the SAS data set being created as seen in this example:

NOTE: The infile '\mysas\recfmvb.dat' is:  
FILENAME=C:\mysas\recfmvb.dat,  
RECFM=V,LRECL=132

NOTE: 5 records were read from the infile '\mysas\recfmvb.dat'.  
The minimum record length was 1.  
The maximum record length was 26.

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

NOTE: The data set WORK.ONE has 2 observations and 5 variables.

NOTE: The DATA statement used 2.85 seconds.

You should have a good idea of what these numbers should be, so a quick check of the NOTES will let you know if there is anything suspicious looking there.

### Use Software Tools to Avoid or Correct Problems

When you are transmitting a file, be sure that the receiving file has the desired characteristics. There are two approaches to use, depending on the type of computer and the software you are using. When using MVS, for example, you can allocate a file, using SPF or a TSO ALLOCATE command to set the file's characteristics, then transmit into that predefined file. Another possibility is overriding the transfer software's default file characteristics. The commands to do this will vary with the software. You will need to use the software's help facility or its documentation to find the proper commands to use. For example the LOCSITE command in FTP is the command to use to specify the record format, logical record length and block size. If you are using 3270 file transfer, there are options in the program for specifying the file characteristics.

You will need a carriage return and line feed at the end of each record when files are transferred from PC to mainframe. If the software you use for the uploading has a menu, look for a place to indicate your preference for this option. Otherwise, use the help facility or software documentation to determine how to specify this option.

SAS options can be used on the INFILE statement to correct problems caused by the default option FLOWOVER. The option used is also a function of the type of input used. The SAS options available are MISSOVER, PAD, TRUNCOVER and STOPOVER. The MISSOVER option prevents a SAS program from going to a new input line, if it does not find values in the current line for all the INPUT statement variables. Missing values are assigned to all variables for which a value was not read. The PAD option controls whether records read from an external file are padded with blanks to the length specified in the LRECL option. NOPAD is the default. The TRUNCOVER option stores whatever data it is able to read for a variable, so you get all the information the record contains. Missing values are assigned to variables for which there is no data. From the basic description it sounds as though MISSOVER and TRUNCOVER accomplish the same thing, which is true, but MISSOVER is only reliable with list input, where TRUNCOVER works for column and formatted input. STOPOVER stops processing the DATA step when an INPUT statement reaches the end of the current record without finding values for all the variables in the statement. The following INFILE statement shows the syntax for using options:

```
INFILE 'filename' option;
```

where *filename* is replaced by the complete name of the file to be read, and *option* is replaced by the option you have chosen to use,

I have taken a very simple data file and read it with all 3 types of input in SAS: list, formatted, and column. With each of the input types I have used the various options that were discussed above. The original data, which is in a file that has RECFM=VB (in TSO) or RECFM=V (in CMS) and LRECL=255, is shown first. You can see what a difference the input type and option used has on the SAS file that is created in the DATA step.

#### ORIGINAL DATA

```

1
21  22
31  32  33
41  42  43  44
51  52  53  54  55

```

INPUT A B C D E; \*For list input the MISSOVER option gives the correct results;

Resulting data set when FLOWOVER, TRUNCOVER or PAD are used:

| OBS | A  | B  | C  | D  | E  |
|-----|----|----|----|----|----|
| 1   | 1  | 21 | 22 | 31 | 32 |
| 2   | 41 | 42 | 43 | 44 | 51 |

Resulting data set when MISSOVER is used:

| OBS | A  | B  | C  | D  | E  |
|-----|----|----|----|----|----|
| 1   | 1  | .  | .  | .  | .  |
| 2   | 21 | 22 | .  | .  | .  |
| 3   | 31 | 32 | 33 | .  | .  |
| 4   | 41 | 42 | 43 | 44 | .  |
| 5   | 51 | 52 | 53 | 54 | 55 |

INPUT (A B C D E) (6.); \*For formatted input the PAD and TRUNCOVER options each produce the correct values;

Resulting data set when FLOWOVER is used:

| OBS | A  | B  | C  | D  | E  |
|-----|----|----|----|----|----|
| 1   | 21 | 31 | 32 | 41 | 42 |

Resulting data set when MISSOVER is used:

| OBS | A  | B  | C  | D  | E |
|-----|----|----|----|----|---|
| 1   | .  | .  | .  | .  | . |
| 2   | 21 | .  | .  | .  | . |
| 3   | 31 | 32 | .  | .  | . |
| 4   | 41 | 42 | 43 | .  | . |
| 5   | 51 | 52 | 53 | 54 | . |

Resulting data set when PAD or TRUNCOVER is used:

| OBS | A  | B  | C  | D  | E  |
|-----|----|----|----|----|----|
| 1   | 1  | .  | .  | .  | .  |
| 2   | 21 | 22 | .  | .  | .  |
| 3   | 31 | 32 | 33 | .  | .  |
| 4   | 41 | 42 | 43 | 44 | .  |
| 5   | 51 | 52 | 53 | 54 | 55 |

INPUT A 1-6 B 7-12 C 13-18 D 19-24 E 25-30;     \*For column input the PAD and TRUNCOVER options both produce the correct results;

Resulting data set when FLOWOVER is used:

No observations in file.

Resulting data set when MISCOVER is used:

| OBS | A  | B  | C  | D  | E |
|-----|----|----|----|----|---|
| 1   | .  | .  | .  | .  | . |
| 2   | 21 | .  | .  | .  | . |
| 3   | 31 | 32 | .  | .  | . |
| 4   | 41 | 42 | 43 | .  | . |
| 5   | 51 | 52 | 53 | 54 | . |

Resulting data set when PAD or TRUNCOVER is used:

| OBS | A  | B  | C  | D  | E  |
|-----|----|----|----|----|----|
| 1   | 1  | .  | .  | .  | .  |
| 2   | 21 | 22 | .  | .  | .  |
| 3   | 31 | 32 | 33 | .  | .  |
| 4   | 41 | 42 | 43 | 44 | .  |
| 5   | 51 | 52 | 53 | 54 | 55 |

There are options available on the INFILE statement that are designed to correct problems related to:

- Delimiters other than a space used.
- Missing values denoted by two adjacent delimiters.
- Delimiters that are part of a character variable.
- Character variables enclosed in quotes.

The option that can be added that will take care of these situations is DSD. It is available in versions 6.07 and later, which means it is not available in the DOS version(6.04). The default delimiter when this option is used is a comma (.). The default treatment of consecutive delimiters is to treat them as a unit, which will cause the INVALIDDATA message. With the DSD option in effect they are treated separately, so a missing value is generated. If a character value is enclosed in quotes and contains the delimiter, this option will cause the quotes to be stripped from the value before it is stored, and any delimiters will be stored with the value. The INFILE statement would be as follows:

```
INFILE 'filename' DSD MISCOVER;
```

The DELIMITER= or DLM= option allows you to specify your own delimiter which will override the default delimiter of a space (a comma if DSD is also used). To specify a comma as a delimiter use the following INFILE statement:

```
INFILE 'filename' DLM=',' MISSOVER;
```

For files that are tab delimited the hexadecimal representation of a tab must be used in the DLM= option in order for SAS to read the files. The following INFILE statement when used on a computer which uses EBCDIC characters (IBM) will read a tab delimited file, recognize two adjacent tabs as representing a missing value, and will keep a tab included in a character variable which is enclosed in quotes :

```
INFILE 'filename' DLM='05'X DSD MISSOVER;
```

Operating systems, such as OpenVMS, DOS, OS/2 and Windows, that use ASCII characters would use the ASCII representation for the tab which is '09'X. The INFILE statement for the DOS version would read:

```
INFILE 'filename' DLM='09'X TABS MISSOVER;
```

The TABS option, which is only available on the DOS version, expands the tab character settings to standard tab settings, which are every 8 columns. NOTABS is the default. From version 6.07 on this option is replaced by EXPANDTABS, and the default is NOEXPANDTABS. You need to remember that the DSD option is not available in the DOS version of SAS, so if missing values are represented by two sequential tabs or if a tab is included in a character variable, the data will not be read correctly. If you are using the Open/VMS, OS/2 or Windows version then add the option DSD to the INFILE statement as shown here:

```
INFILE 'filename' DLM='09'X DSD MISSOVER;
```

When ASCII files are produced that contain the names of the variables in the first record the potential problem can be eliminated by using an editor to delete that record. You can also skip over those records by using the FIRSTOBS= option on the INFILE statement. For example to skip over the variable names on the first record of a file use the following statement:

```
INFILE 'filename' FIRSTOBS=2 options;
```

where *options* is replaced by any other INFILE options you want to use. FIRSTOBS= tells the INPUT statement which record it should read as the first observation. At first glance this looks like a good solution, but if this option is used on an OPTIONS statement instead of the INFILE statement subsequent data set references are affected. The options listed on the OPTIONS statement are global, rather than local as on the INFILE statement. I was reminded of this as I puzzled over a program that seemed to lose one observation every time it was referenced in the program.

To override the default logical record length of 132, in the PC-based versions of SAS use the LRECL= option on the INFILE statement as follows:

```
INFILE 'filename' LRECL=xxx options;
```

where *xxx* is replaced with the length (maximum length if the records are of variable length) of the records you are reading, and *options* is replaced by any additional INFILE options you want to use.

On TSO when unnumbered files are copied without using the NONUM option, 0's are put in columns 73-80. The proper form of the COPY command is the following:

```
COPY oldfilename newfilename NONUM
```

If the file is a SAS program, when SAS tries to compile the program it attempts to translate the information in columns 73-80 as part of the SAS code. SAS determines if a file is to be treated as numbered (columns 73-80 are ignored) or unnumbered (columns 73-80 are processed) by checking the contents of columns 73-80. If these columns contain sequential numbers, the file is treated as numbered, otherwise it is considered an unnumbered file. Since these columns do not appear on most monitors unless you scroll the line, your first indication of a problem may be when you get an error on virtually every line in your program. Columns 73-80 will be flagged as the offending columns in this case, but again, if you are looking at the SAS log on your monitor, the columns will not show up until you scroll the line. This is a tricky problem to find sometimes, but it is generally easy to correct. Use your editor to change all of the occurrences of a string of 0's in columns 73-80 to blanks, then it is an unnumbered data set. If it is a data file that was copied without the NONUM option, and you are using list input in SAS to read the file, the program may try to process the zeroes in columns 73-80 as part of the data.

### **Check the Results of the INPUT Statement**

You should always use the PRINT procedure to print at least a few observations from the SAS data set that you create using the INPUT statement and check it against the original data. If the original data and the values read into the SAS variables are correct, you can be fairly confident that you are starting with good data rather than garbage.

### **Conclusion**

This paper examined specific problems caused by data transfer (PC-based systems to MVS, CMS and OpenVMS) and data entry. Indicators of problems that could be detected by visual data checking and from the NOTES in the SAS log were given. The use of options for defining such things as record format, logical record length, blocksize and carriage return, line feed in data transfer software; options on the SAS INFILE statement including MISCOVER, TRUNCOVER, STOPOVER, PAD, DSD, DLM and LRECL; and the TSO commands ALLOCATE and COPY with the NONUM option were discussed. The PRINT procedure was used to determine if all the preventive and/or corrective measures had been effective. With these tools added to your toolbox you can avoid being taken by surprise by problems generated during data transfer and data entry.

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA