Data Step Information 3 – INFILE Statement

Data measurements do not have to be included as part of the SAS program. Data may be saved in separate files and read into a SAS program. There are several ways to read in a data file.

INFILE statement

When the data is stored in a *text* file (*.txt, *.dat, or other extension), it may be read into a SAS program using an INFILE statement. When an INFILE statement is used in a SAS program, it must follow the DATA statement and come before the INPUT statement. When an INFILE statement is used, no DATALINES (or CARDS) statement is needed. Data transformations or variable assignment statements can still be a part of the program. As before, these statements always follow the INPUT statement.

When using an INFILE statement, you must know the variable names and the order in which they occur in the data set. The text files can be data only, or they may contain column headers or other information at the top of the file.

An INFILE statement can be used in conjunction with a FILENAME statement. The FILENAME statement appears before the DATA statement. The purpose of the FILENAME statement is to give the location (directory information – drive:\folder) of the data file and assign a temporary or convenience name to the file that is to be read. Only one INFILE statement is permitted with each DATA statement.

The syntax of the FILENAME and INFILE statements:

```
FILENAME  tempname  'location and name of file' ;
DATA   tablename;
INFILE  tempname  <options> ;
INPUT  variable1 ... variablen;
data transformations - if any
```

The FILENAME statement is not necessary if the complete file path is specified in the INFILE statement. The filename and path must be enclosed in quotation marks in either case.

A few of the options to be used in the INFILE statement are:

DELIMITER = *delimiters*
DLM = *delimiters*

Delimiters are the symbols or characters used to separate pieces of information in a data file. Up to this point, items in a data file have been separated by at least one space. When no spaces separated the data, column formatting in the INPUT statement should be used. A delimiter can be a comma (,), a tab, a space, a letter, or a combination of letters. When the delimiter is a space, it is not necessary to use a DELIMITER option.
For example, consider a comma delimited file called temp.dat stored in the drive or folder identified in this illustration as a:\

```
75,6,120
219,14,101
```

To read in a file such as this:

```
DATA one;
INFILE 'a:\temp.dat' DLM = ',' ;
INPUT x y z;
RUN;
```

(Note that in this example a FILENAME statement is not used, and the complete file path and name are specified in quotation marks in the INFILE statement.)

DSD

requests that two adjacent delimiters indicate a missing observation. For example, if a line in the external data file is 75,,120 and the delimiter is a comma, a missing observation ('.') will be recorded "between" 75 and 120.

FIRSTOBS = *record-number*

indicates that you want to begin reading the input file at the record number specified, rather than beginning with the first record. This allows you to store information other than the data 'at the top' of an external data file. For example,

```
INFILE datatable  FIRSTOBS = 10;
```

specifies that the reading of the data begins at line 10 of the external file.

LRECL = *record-length*

Specifies the record length in bytes. The default is 256. Values between 1 and 1,048,576 (1 megabyte) can be specified. This is useful for external data files that are "wide," that is, contain a large number of columns.

MISSOVER

prevents a SAS program from going to a new input line if, when using list input, it does not find values in the current line for all the INPUT statement variables. When an INPUT statement reaches the end of the current record, values that are expected but not found are set to missing.

OBS = *record-number*

specifies the record number of the last record that you want to read from an input file that is being read sequentially. For example,

```
INFILE dataset  OBS = 30;
```

indicates that the last record from the external file to be read is the 30th item in the file.

STOPOVER

stops processing the DATA step when an INPUT statement using list input reaches the end of the current record without finding values for all variables in the statement.

Objective 1:

In the SAS Editor, enter the following data exactly as it appears below. Do not use the tab key. Insert at least one space between the entries in a single row. (TAB has a different effect than space. TAB is covered on the next page.)

```
1998     7     35
1999     8     54
2000    11    101
```

Save this file as *tornado1.dat* by selecting **File-Save as-Save as type-Data files & enter file name in the blank**. Note the drive:\folder in which you've saved the file Modify the above text so that it appears exactly as it appears below. (No TABS, space bar only.)

```
YEAR   NUM    DAMAGE
1998     7     35
1999     8     54
2000    11    101
```

Save this file as *tornado2.dat*. Modify the text one more time so that it appears exactly as it appears below.

```
1998 7 35
1999 8 54
200011101
```

Save this file as *tornado3.dat*. Clear the text in the editor window. Enter the following SAS program. You will want to specify the drive:\folder in which you've saved the files rather than a: in the program below.

```
FILENAME  t1  'a:\tornado1.dat' ;
FILENAME  t2  'a:\tornado2.dat' ;
FILENAME  t3  'a:\tornado3.dat' ;

DATA one;
INFILE  t1;
INPUT  YEAR  NUMBER  DAMAGE;
PROC PRINT DATA=ONE;
TITLE 'Objective 1';

DATA two;
INFILE t2  FIRSTOBS=2 ;
INPUT year number damage ;;
PROC PRINT DATA=two;
```

```
DATA three;
INFILE  t3;
INPUT Year 1-4  Number 5-6  Damage 7-9;
PROC PRINT DATA=three;
RUN;
QUIT;
```

The printed output of these three files should be identical with the exception of the column headings (variable names). In the output window, notice that the uppercase and lowercase letters you use appear exactly as you typed them in the INPUT statement. When reading in any data set using the INFILE statement, be certain to check the LOG window for the appropriate number of observations in the data set. Note that data table three requires you to use column formatting in conjunction with the INFILE statement.

As stated above, the FILENAME statement is not necessary. To read in the data file *tornado1.dat*, the following commands can also be used.

```
DATA one;
INFILE 'a:\tornado1.dat';
INPUT year number damage;
RUN;
```

CAUTION: Text files of data do not have to be created in the SAS Editor. They can be created using other software. They must be saved as a text file though. Depending upon the software used to create these files, the file extensions could be quite different. A couple of common extensions are .txt and .prn. Many spreadsheets have the option of saving the file as a tab delimited file. You can import these into SAS also. For files that are tab delimited:

```
FILENAME adata 'a:\temp.dat';
DATA one;
INFILE adata DLM = '09'X DSD;                  '09'X is the hexadecimal code for tab
                          DSD indicates that two adjacent tabs will imply that a value is missing
INPUT x y z;
RUN;
```

Objective 2: In the Editor, open tornado3.dat. Insert tabs between variable values – two tabs on each line. Save this tab delimited file as tornado4.dat. Write the SAS code you would use to read and print the contents of this data table. This time, do not use a FILENAME statement.

Text files of data are a convenience for files that are particularly large or that will require long-term statistical analysis. Data text files are also convenient when data sets are to be given to several people on a storage device, Internet web sites, or e-mail.

Other SAS procedures exist that read in data tables that includes the variable names. See PROC IMPORT, PROC DBF, and DDE for related information.