INPUTTING DATA INTO SAS | SAS LEARNING MODULES

This module will show how to input raw data into SAS, showing how to read instream data and external raw data files using some common raw data formats. Section 3 shows how to read external raw data files on a PC, UNIX/AIX, and Macintosh, while sections 4-6 give examples showing how to read the external raw data files on a PC, however these examples are easily converted to work on UNIX/AIX or a Macintosh based on the examples shown in section 3.

1. Reading free formatted data instream

One of the most common ways to read data into SAS is by reading the data instream in a **data step** – that is, by typing the data directly into the syntax of your SAS program. This approach is good for relatively small datasets. Spaces are usually used to "delimit" (or separate) free formatted data. For example:

```
DATA cars1;
 INPUT make $ model $ mpg weight price;
CARDS;
AMC Concord 22 2930 4099
AMC Pacer   17 3350 4749
AMC Spirit  22 2640 3799
Buick Century 20 3250 4816
Buick Electra 15 4080 7827
;
RUN;
```

After reading in the data with a data step, it is usually a good idea to print the first few cases of your dataset to check that things were read correctly.

```
title "cars1 data";
PROC PRINT DATA=cars1(obs=5);
RUN;
```

Here is the output produced by the **proc print** statement above.

```
cars1 data

OBS    MAKE       MODEL      MPG    WEIGHT    PRICE
1      AMC        Concord     22      2930      4099
2      AMC        Pacer       17      3350      4749
3      AMC        Spirit      22      2640      3799
4      Buick      Century     20      3250      4816
5      Buick      Electra     15      4080      7827
```

2. Reading fixed formatted data instream

Fixed formatted data can also be read instream. Usually, because there are no delimiters (such as spaces, commas, or tabs) to separate fixed formatted data, column definitions are required for every variable in the dataset. That is, you need to provide the beginning and ending column numbers for each variable. This also requires the data to be in the same columns for each case. For example, if we rearrange the cars data from above, we can read it as fixed formatted data:

```
DATA cars2;
  INPUT make $ 1-5 model $ 6-12 mpg 13-14 weight 15-18 price 19-22
CARDS;
AMC  Concord2229304099
AMC  Pacer  1733504749
AMC  Spirit 2226403799
BuickCentury2032504816
BuickElectra1540807827
;
RUN;

TITLE "cars2 data";
PROC PRINT DATA=cars2(obs=5);
RUN;
```

The benefit of fixed formatted data is that you can fit more information on a line when you do not use delimiters such as spaces or commas.

Here is the output produced by the **proc print** statement above.

```
cars2 data

OBS     MAKE       MODEL      MPG    WEIGHT    PRICE
1       AMC        Concord     22      2930      4099
2       AMC        Pacer       17      3350      4749
3       AMC        Spirit      22      2640      3799
4       Buick      Century     20      3250      4816
5       Buick      Electra     15      4080      7827
```

3. Reading fixed formatted data from an external file

Suppose you are using a PC and you have a file named **cars3.dat**, that is stored in the **c:carsdata** directory of your computer. Here's what the data in the file **cars3.dat** look like:

```
AMC  Concord2229304099
AMC  Pacer  1733504749
AMC  Spirit 2226403799
BuickCentury2032504816
BuickElectra1540807827
```

To read the file **cars3.dat**, use the following syntax.

```
 DATA cars3;
  INFILE "c:carsdatacars3.dat";
  INPUT make $ 1-5 model $ 6-12 mpg 13-14 weight 15-18 price 19-2
RUN;

TITLE "cars3 data";
PROC PRINT DATA=cars3(obs=5);
RUN;
```

Here is the output produced by the **proc print** statement above.

```
cars3 data

OBS    MAKE      MODEL     MPG    WEIGHT    PRICE
1      AMC       Concord    22      2930      4099
2      AMC       Pacer      17      3350      4749
3      AMC       Spirit     22      2640      3799
4      Buick     Century    20      3250      4816
5      Buick     Electra    15      4080      7827
```

Suppose you were working on UNIX. The UNIX version of this program, assuming the file **cars3.dat** is located in the directory **~/carsdata**, would use the syntax shown below. (Note that the "**~**" in the UNIX pathname above refers to the user's HOME directory. Hence, the directory called **carsdata** that is located in the users HOME directory.)

```
DATA cars3;
  INFILE "~/carsdata/cars3.dat";
  INPUT make $ 1-5 model $ 6-12 mpg 13-14 weight 15-18 price 19-2
RUN;

TITLE "cars3 data";
PROC PRINT DATA=cars3(obs=5);
RUN;
```

Likewise, suppose you were working on a Macintosh. The Macintosh version of this program, assuming **cars3.dat** is located on your hard drive (called **Hard Drive**) in a folder called **carsdata** would look like this.

```
DATA cars3;
  INFILE 'Hard Drive:carsdata:cars3.dat';
  INPUT make $ 1-5 model $ 6-12 mpg 13-14 weight 15-18 price 19-2
RUN;

TITLE "cars3 data";
PROC PRINT DATA=cars3(OBS=5);
RUN;
```

In examples 4, 5 and 6 below, you can change the **infile** statement as these examples have shown to make the programs appropriate for UNIX or for the Macintosh.

4. Reading free formatted (space delimited) data from an external file

Free formatted data that is **space** delimited can also be read from an external file. For example, suppose you have a space delimited file named **cars4.dat**, that is stored in the **c:carsdata** directory of your computer.

Here's what the data in the file **cars4.dat** look like:

```
AMC Concord 22 2930 4099
AMC Pacer   17 3350 4749
AMC Spirit  22 2640 3799
Buick Century 20 3250 4816
Buick Electra 15 4080 7827
```

To read the data from **cars4.dat** into SAS, use the following syntax:

```
DATA cars4;
  INFILE "c:carsdatacars4.dat";
  INPUT make $ model $ mpg weight price;
RUN;

TITLE "cars4 data";
PROC PRINT DATA=cars4(OBS=5);
RUN;
```

Here is the output produced by the **proc print** statement above.

```
cars4 data

OBS    MAKE      MODEL     MPG    WEIGHT    PRICE
1      AMC       Concord   22     2930      4099
2      AMC       Pacer     17     3350      4749
3      AMC       Spirit    22     2640      3799
4      Buick     Century   20     3250      4816
5      Buick     Electra   15     4080      7827
```

5. Reading free formatted (comma delimited) data from an external file

Free formatted data that is **comma** delimited can also be read from an external file. For example, suppose you have a comma delimited file named **cars5.dat**, that is stored in the **c:carsdata** directory of your computer.

Here's what the data in the file **cars5.dat** look like:

```
AMC,Concord,22,2930,4099
AMC,Pacer,17,3350,4749
AMC,Spirit,22,2640,3799
Buick,Century,20,3250,4816
Buick,Electra,15,4080,7827
```

To read the data from **cars5.dat** into SAS, use the following syntax:

```
DATA cars5;
  INFILE "c:carsdatacars5.dat" delimiter=',';
  INPUT make $ model $ mpg weight price;
RUN;

TITLE "cars5 data";
PROC PRINT DATA=cars5(OBS=5);
RUN;
```

Here is the output produced by the **proc print** statement above.

```
cars5 data

OBS    MAKE      MODEL     MPG    WEIGHT    PRICE
1      AMC       Concord   22     2930      4099
2      AMC       Pacer     17     3350      4749
3      AMC       Spirit    22     2640      3799
4      Buick     Century   20     3250      4816
5      Buick     Electra   15     4080      7827
```

6. Reading free formatted (tab delimited) data from an external file

Free formatted data that is **TAB** delimited can also be read from an external file. For example, suppose you have a tab delimited file named **cars6.dat**, that is stored in the **c:carsdata** directory of your computer.

Here's what the data in the file **cars6.dat** look like:

```
AMC     Concord 22      2930    4099
AMC     Pacer   17      3350    4749
AMC     Spirit  22      2640    3799
Buick   Century 20      3250    4816
Buick   Electra 15      4080    7827
```

To read the data from **cars6.dat** into SAS, use the following syntax:

```
DATA cars6;
  INFILE "c:carsdatacars6.dat" DELIMITER='09'x;
  INPUT make $ model $ mpg weight price;
RUN;

TITLE "cars6 data";
PROC PRINT DATA=cars6(OBS=5);
RUN;
```

Here is the output produced by the **proc print** statement above.

```
cars6 data

OBS    MAKE      MODEL     MPG    WEIGHT    PRICE
1      AMC       Concord   22     2930      4099
2      AMC       Pacer     17     3350      4749
3      AMC       Spirit    22     2640      3799
4      Buick     Century   20     3250      4816
5      Buick     Electra   15     4080      7827
```

7. Problems to look out for

- If you read a file that is wider than 80 columns, you may need to use the **lrecl**= parameter on the **infile** statement.

8. For more information

- For more detailed information on reading raw data into SAS, see Reading data into SAS (/sas/library/sas-libraryreading-data-into-sas/) in the SAS Library (/sas/library/).
- To learn how to create permanent SAS system files, see the Reading and writing SAS system files (https://stats.idre.ucla.edu/sas/modules/making-and-using-permanent-sas-data-files-version-8/).
- For information on creating and recoding variables once you have entered your data, see the SAS Learning Module (https://stats.idre.ucla.edu/sas/modules/) on Creating and recoding variables (/sas/modules/creating-and-recoding-variables-in-sas/).

Click here to report an error on this page or leave a comment

How to cite this page (https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-cite-web-pages-and-programs-from-the-ucla-statistical-consulting-group/)