

Data Step

El Data Step consiste de una serie de instrucciones que crean un data set que SAS puede analizar.

Para crear este data set podemos leer los datos de:

- Directamente en el programa
- Archivo no procesado (texto, ASCII)
- Archivo SAS
- Otros programas como Excel, dBase, Oracle
-

Leer los datos directamente en el programa

```
DATA estudiantes;  
INPUT id $ apellido $ nombre $ promedio;  
DATALINES;  
801-45-1225 García José 80  
801-25-3678 Rivera Juan 84  
RUN;
```

Cada línea de datos consiste de 80 columnas a no ser que usemos la opción de NOCARDIMAGE

```
PROC OPTIONS GROUP=nocardimage;
```

Crea el archivo temporero *estudiantes* que contiene 2 observaciones y 4 variables (1 numérica y 3 caracteres)

Leer los datos de un archivo no procesado

```
DATA estudiantes;  
INFILE 'c:\sas\datos\reg.txt';  
INPUT id $ apellido $ nombre $ promedio;  
RUN;
```

Crea el archivo temporero *estudiantes*, en la librería WORK, que contiene los datos que están el archivo reg.txt en la librería (directorio) c:\sas\datos. Los datos de este archivo (reg.txt) están separados por uno o más espacios y hay un record (id, nombre, apellido y promedio de un estudiante) por línea.

Si cambiamos la primera instrucción por DATA 'estudiantes'; se crea un archivo permanente en la librería SASUSER.

INFILE nombre opciones;

nombre:

- *'path/archivo_externo'*
- *fileref* referencia a un archivo externo, previamente asignado usando la instrucción FILENAME *fileref* *'path/archivo_externo'*;
- DATALINES – especifica que los datos están después de la instrucción DATALINES en el DATA step. Esto permite usar opciones de INFILE cuando se leen los datos con DATALINES.

Opciones (algunas):

- obs=n lee los primeros n records
- firstobs=k comienza a leer en el record #k hasta el final
- lrecl=l largo del record en el archivo. El default es 256.
- missover impide que la instrucción de INPUT lea un record nuevo si no encuentra valores para todas las variables en la línea que está leyendo. Cuando llega al final del record y no encuentra más datos, a las variables que se quedaron sin valores SAS les asigna "." o espacio (valores perdidos para variables numéricos o caracteres).
- truncover cuando leemos usando column o formatted input y algunas líneas de datos son más cortas que otras. Indica a SAS que siga leyendo el valor de la variable hasta que llegue al final de la línea o a la última columna especificada por el alcance del formato o columna.
- dlim='c' delimiter-indica que los datos están separados por este carácter. El default es espacio.
 dlim=', ' los datos están separados por comas

 dlim='09'X los datos están separados por TAB

 09 es el código ASCII de TAB en hexadecimal y X indica que está en hexadecimal
- dsl - Indica que los datos están separados por comas.
 - Ignora los separadores que están entre comillas
 - No lee las comillas como parte del valor del dato.
 - Considera dos separadores seguidos como un "missing value"
 - Si el separador no es una coma, se puede usar la opción DLM= con la opción DSD para especificar el separador

Ejemplos

1. Para leer un archivo ASCII con “missing values” donde el separador es un TAB usaríamos la siguiente instrucción:

```
INFILE 'nombre' DLM='09'x DSD;
```

2. Para usar ‘,’ como separador al usar DATALINES:

```
DATA 'datos';  
INFILE DATALINES DLM=',';  
INPUT nombre $ apellido $;  
DATALINES;  
Juan,Garcia  
Pedro,Rivera  
RUN;
```

3. Para leer los siguientes datos:

```
Juan Garcia    215  Calle 4  
Silvia Alonso 1524 Ave Ponce de Leon
```

```
DATA direccion;  
INFILE 'c:\nombre.txt' TRUNCOVER;  
INPUT name $ 1-15 num 16-19 calle $ 23-38;  
RUN;
```

INPUT

Asigna nombres a las variables e indica en que posición están.

Esta instrucción depende de cómo estén los datos en el archivos:

List input y List input modificado

- datos están separados por espacio(s) o un carácter definido por DLM.
- Cada línea representa un record.
- En el caso de variables carácter sus valores deben tener 8 caracteres o menos, sino no los lee correctamente.
- El modificado es cuando se usan cierto formatos (&, :, ~).

```
INPUT <pointer> var <$> <&>;
```

```
INPUT <pointer> var <:|&|~> <informat.> <@|@@>;
```

(a|b|c indica que es a, b o c; <a> indica que a es opcional)

Pointers:

- @n apunta a la columna n

- @'char_str' busca *char_str* en el record y apunta a la primera columna después de *char_str*.

- #n apunta a record (línea) n

- +n mueve el puntero n columnas.

- / mueve el puntero a la columna 1 de la línea siguiente.

Ejemplos:

```
INPUT @3 nota +2 prom;
```

lee la variable nota comenzando en la columna 3, se mueve 2 espacios y lee el promedio.

```
INPUT name age #3 id;
```

lee las variables name y age de la primera fila y id de la tercera.

```
INPUT name age / id;
```

lee las variables name y age de la primera fila y id de la segunda.

\$, &, :, ~

\$ indica que la variable a su izquierda es una variable carácter

& indica que la variable carácter tiene uno o más espacios simples en su valor. Para indicar dónde termina el valor de la variable hay que poner dos espacios seguidos, de lo contrario sigue incluyendo todo hasta el final de la línea o hasta el largo de la variable si este fue definido.

```
INPUT nombre $ & nota ;  
DATALINES;  
A. I. C. 100  
Gabriela 80  
RUN;
```

lee el nombre, A. I. C., hasta que encuentra dos espacios corridos, luego lee nota, 100.

: se usa con INFORMATS pero los datos no están en columnas.

```
INPUT nombre : $10. nota 3.0;
DATALINES;
Ada 100
Gabriela 80
RUN;
```

lee el nombre, Ada, hasta que encuentra el primer espacio, luego lee nota, 100.

~ lee como carácter lo que está entre comillas simples o dobles y mantiene las comillas.

Hay que usarlo con la opción DSD del INFILE.

```
INFILE reg DSD;
INPUT nombre ~ $25. nota;
DATALINES;
"Ada Ray",100
RUN;
```

lee el nombre, "Ada Ray", luego lee nota, 100.

Line-hold specifier

@ trailing @-le dice a SAS que se quede en la línea que está leyendo y no pase a la siguiente. Es útil cuando queremos leer varias veces de la misma línea. Para pasar a la próxima línea hay que poner un INPUT sin @.

```
DATA test;
INFILE 'c:\SAS\trafico.dat';
INPUT tipo $ @;
IF tipo='carretera' THEN DELETE;
INPUT nombre $ 9-38 am pm;
```

@@ double trailing @- le dice a SAS que se quede en la línea que está leyendo y no pase a la siguiente. Es útil hay varios records por línea. Para pasar a la próxima línea hay que poner un INPUT sin @@ o que llegue al final de la línea..

Column input

- datos en columnas.
- Se pueden leer las variables en cualquier orden y saltar algunas
- variables carácter pueden incluir espacios.
- Variables numéricas contienen números, puntos decimales, +, -, E

```
INPUT var <$> startcol-endcol <.decimales> <@|@@>;
```

Starcol – primera columna que tiene el valor que vamos a leer

Endcol – última columna que tiene el valor que vamos a leer

.decimales – cantidad de decimales que tiene el número cuando éste no tiene explícitamente el punto decimal. Si el número tiene el punto decimal este prevalece sobre el otro.

Ejemplo:

```
INPUT nombre $ 1-10 salario 30-35 .2 apellido $ 12-24;
```

Lee el *nombre* que se encuentra en las columnas 1 a la 10, luego *salario* en las columnas de la 30 a la 35 y pone un punto decimal antes del penúltimo número, si el número no tiene ningún punto, de lo contrario ignora la opción de decimales en el INPUT. Por último lee el *apellido* en las columnas 12 a la 24. Nombre y apellido son variable carácter y salario es numérica.

Formatted input

- datos están en un formato no estándar. (Por ejemplo: \$100,000, fechas)
- Usa INFORMATS para decirle a la computadora como leer estos datos.

```
INPUT <pointer> var informat. <@|@@>;  
INPUT <pointer> (var-list) (informat-list) <@|@@>;  
INPUT <pointer> (var-list) (<n*>informat.) <@|@@>;
```

n* - especifica que se repita n veces el informat que le sigue.

Informats

\$informat-namew.d

\$	indica una variable carácter
w	largo del campo (opcional)
d	decimales (opcional)

informats comunes:

- \$CHARw.** Lee variable carácter de largo w sin eliminar los espacios de al principio o final. Por default w es 8.
- \$w.** Lee variable carácter de largo y elimina los espacios del principio. Hay que especificar w.
- DATEw.** lee fechas de la forma ddmmyy o ddmmyyyy (ej. 30jan2003) Por default w es 7
- MMDDYYw.** lee fechas de la forma ddmmyy o ddmmyyyy (ej.01/30/2003 ó 01-30-03). Por default w es 6.
- COMMAw.d** remueve comas y \$ incluidos. Convierte paréntesis izquierdo en signo de menos. Por default w es 1.
- PERCENTw.** convierte por cientos a números. Por default w es 6.
- w.d** lee daros numéricos estándar de largo w con d decimales. Hay que especificar w. Si el número ya tiene un punto decimal se ignora d.

Si los valores de una variable leída con un informat no son todos del mismo largo (w), las variables siguientes no se leen correctamente.

Si el valor (dato) tiene un largo mayor que w solo se leerán los primeros w (si tiene punto decimal, coma, signo de dólar, etc. estos cuentan también)

Para evitar lo anterior, en caso de ser necesario, usamos la instrucción:

INFORMAT *var-name informat;*

antes de la instrucción INPUT. De esta forma se comporta como si se usará la opción de : en el INPUT e ignora w.

num	INPUT num 6.2;	INFORMAT num 6.2; INPUT num;
12345678	1234.56	123456.78

Ejemplos:

Valor	Informat	Valor leído
10/29/99	MMDDYY8.	14546 (número de días entre ene. 1, 1960 y oct. 29, 1999)
125.83	4.2	125.00 (truncates)
1234	5.1	123.4
2.25	4.2	2.25
ESTA 4206	\$12.	ESTA 4206
ESTA 4206	\$CHAR12.	ESTA 4206
\$1,234.25	Comma9.2	1234.25

Entrada de datos con la ventana de ViewTable

- Manera fácil de crear un Data set de SAS
- Se encuentra en TOOLS→Table Editor
- Las filas son los records y columnas son las variables
- Right-click en la letra que está arriba de una columna para poner los atributos (nombre, tipo, largo, etc.) de la variable.
- Guardarla con File →Save As
- Para abrir y/o editar una tabla:
 - Abrir Table Editor
 - Seleccionar File →Open
- Para imprimir los datos:

PROC PRINT DATA=*libref.name*;

Import y Export Wizard

Podemos importar un archivo creado en otros programas como Excel, dBase, Oracle, Acces, etc..

Podemos exportar un archivo SAS a un file de texto Tab-delimited, Excel, Access, etc.

Otra forma de exportar un archivo SAS: FILE & PUT

```
DATA _NULL_;  
SET sasuser.mydata;  
FILE 'c:\temp\newfile';  
PUT curso 'SAS'@25 nombre $20.;
```

NULL evita que se cree un data set
SET lee un archivo permanente de SAS
FILE dice el nombre y lugar donde guardar el archivo nuevo;
PUT especifica qué guarda y donde. Lo que está entre comillas SAS lo inserta en el file.