

## Manejo de archivos

### Leer un data set de SAS

```
DATA data-set-nuevo (opciones-data-sets);  
  SET data-set-SAS (opciones-data-sets);
```

Opciones-data-sets:

Se usan en DATA step y PROC pero al leer el data set

FIRSTOBS=*n* – empieza a leer en la observación *n*.  
OBS=*n* – termina de leer en la observación *n*.

Afectan las variables que se van a guardar o leer, dependiendo de donde estén, ej. en DATA afecta las variables que se guardan al archivo y en SET las que se van a leer.

KEEP=*var-list* – lista de variables que se van a incluir en el archivo Nuevo.  
DROP= *var-list* – lista de variables que no se van a incluir en el archivo nuevo.  
RENAME=(*old-var-name=new-var-name*)

Se usa en SET, MERGE y UPDATE

IN=*var*; – *var* tiene un valor de uno si el data set contribuyó a la observación en proceso y cero si no.

Si estamos usando la opción de KEEP y creamos un nueva variable, esta debe incluirse también en el KEEP o no se guardará en el archivo creado.

También hay KEEP y DROP statements que se pueden usar solo en el DATA step para especificar qué variables se van a guardar. En este caso afectan a todos los data sets creados en el DATA statement.

### “Stacking” data sets

```
DATA name;  
  SET data1 data2 ... datan;
```

Pone las observaciones del data set2 debajo del data set1, el data set3 debajo del data set2 y así sucesivamente.

El total de observaciones es la suma de la cantidad de observaciones en cada set.

El total de variables es el total de variables en común más el total de variables diferentes.

## Intercalando data sets

```
DATA data-set-nuevo;  
  SET data1 data2 ... datan;  
  BY var-list;
```

Une data sets pero en lugar de poner uno debajo del otro, intercala las observaciones de forma que el data set nuevo queda ordenado por *var-list*. Para esto los data sets originales tienen que estar ordenados por *var-list*..

## Combinado data sets: MERGE

```
DATA comb;  
  MERGE data-set1 data-set2;  
  BY var-list;
```

Ambos archivos tienen que estar ordenados con anterioridad por las variables en BY var-list.

```
DATA ejemplo;  
  MERGE set1 (IN=e1) set2 (IN=e2);  
  BY num;  
  IF ~(e1=1 & e2=1); *no incluye los repetidos;  
  PROC PRINT;  
  RUN;
```

Si no ponemos ninguna variable en el BY statement, SAS combina la primera observación del data-set-1 con la primera del data-set-2, la segunda con la segunda y así sucesivamente. Si hay variables con el mismo nombre los valores del data-set-2 reemplazan los valores del data-set-1. Esto puede tener resultados no deseados.

One-to-one merge – cuando no ponemos el BY statement.

cuenta	mes		cuenta	Ventas		cuenta	mes	Ventas
2-301	ene	+	2-290	25	→	2-290	ene	25
2-401	mar		2-401	35		2-401	mar	35
2-420	abr		2-420	15		2-420	abr	15

One- to many merge cuando usamos el BY statement

cuenta	mes		cuenta	Ventas		cuenta	mes	Ventas
2-301	ene	+	2-401	25	→	2-301	ene	.
2-401	mar		2-401	35		2-401	mar	25
2-420	abr		2-420	15		2-401	mar	35
						2-420	abr	15

Many-to-many (usamos BY statement)

cuenta	mes		cuenta	Ventas		cuenta	mes	Ventas
2-301	ene	+	2-301	12	→	2-301	ene	12
2-401	mar		2-301	25		2-301	ene	25
2-401	abr		2-401	35		2-401	mar	35
2-501	may		2-401	15		2-401	abr	15
						2-501	may	.

### Combinando estadísticos con data set original

```

DATA banco;
INPUT cuenta $ trans @@;
CARDS;
1-23 10 2-54 5 3-62 6 1-23 3 3-62 1 1-23 3 2-54 7
RUN;
PROC SORT DATA=banco out=bank;
  BY cuenta;

/* Guarda total de transacciones por cuenta*/
PROC MEANS NOPRINT DATA=bank;
  VAR trans;
  BY cuenta;
  OUTPUT OUT=resumen SUM(trans)=total;
PROC PRINT DATA=resumen;
TITLE 'Resumen de los datos';
DATA conjunto;
  MERGE bank resumen;
  BY cuenta;
  DROP _type_ _freq_;
  Percent=trans/total*100;
PROC PRINT DATA=conjunto NOOBS;
  BY cuenta;
  ID cuenta;
  VAR trans total percent;
  TITLE2 'Por cientos';
RUN;

```

## Combinando un total con data set original

```
DATA data-set-nuevo;  
  IF _N_=1 THEN SET data-set-con-totales;  
  SET data-set-original;
```

\_N\_ - número de veces que SAS a pasado a través del DATA step.

```
DATA banco;  
INPUT cuenta $ trans @@;  
CARDS;  
1-23 10 2-54 5 3-62 6 1-23 3 3-62 1 1-23 3 2-54 7  
RUN;  
  
/* Guarda total de transacciones*/  
PROC MEANS NOPRINT DATA=banco;  
  VAR trans;  
  OUTPUT OUT=resumen SUM(trans)=total;  
  
PROC PRINT DATA=resumen;  
  TITLE 'Resumen de los datos';  
  
/* Combina los dos archivos: resumen y banco*/  
DATA conjunto;  
  IF _N_=1 THEN SET resumen;  
  SET banco;  
  Percent=trans/total*100;  
PROC SORT;  
  BY cuenta;  
PROC PRINT DATA=conjunto NOOBS;  
  VAR trans total percent;  
  BY cuenta;  
  TITLE2 'Por cientos';  
RUN;
```

## Actualización de un data set

```
DATA master;  
  UPDATE master transact;  
  By var-list;
```

UPDATE – combina dos data sets comparando observaciones de variables comunes. Ambos archivos deben estar ordenados por las variables en *var-list*.

Algunas diferencias entre UPDATE con MERGE y MODIFY:

- el archivo *master* nuevo tiene una sola observación para cada valor de las variables comunes.
- Missing values en el archivo *transact* no cambian los valores en el *master*
- Se pueden añadir nuevas variables.

```
* Calcula las transacciones totales por cuenta y las guarda en balances;
DATA transacciones;
INFILE 'c:\temp\tran.txt';
INPUT cuenta $ trans;
RUN;
PROC SORT DATA=transacciones;
BY cuenta;
PROC MEANS DATA=transacciones NOPRINT;
VAR trans;
BY cuenta;
OUTPUT OUT=balances SUM(trans)=trans;

* Actualiza las transacciones en el archivo master Banco y calcula el nuevo balance;
DATA banco;
UPDATE banco balances;
BY cuenta;
balance=balance-trans;
trans=0;
DROP _type_ _freq_;
PROC PRINT DATA=banco;
RUN;
```

## OUTPUT statement

OUTPUT *data-set*;

El OUTPUT statement hace que SAS grabe la observación actual al archivo *data-set* inmediatamente, no al final del DATA step. Todo DATA step tiene un OUTPUT statement implícito al final. Si usamos un OUTPUT statement explícitamente controlamos dónde se graban las observaciones y cuándo.

Podemos crear varios data sets a la vez si ponemos varios nombres en el DATA statement: DATA set-1 set-2 ...;

Si escribimos OUTPUT; (sin nombre de archivo) guarda la misma observación a todos los archivos del DATA statement.

El nombre del archivo que ponemos en OUTPUT statement tiene que estar también en el DATA statement.

Podemos dividir un archivo (curso.txt) en varios archivos (seccion\_1 y seccion\_2).

```
DATA seccion_1 seccion_2;
INFILE 'c:\temp\curso.txt';
INPUT nombre $25. ex1-ex4 sec $;
IF sec='01' THEN OUTPUT seccion_1;
      ELSE OUTPUT seccion_2;
PROC PRINT DATA =seccion_1;
TITLE 'Sección 1';
PROC PRINT DATA =seccion_2;
TITLE 'Sección 2';
RUN;
```

También podemos utilizar el OUTPUT statement para crear varias observaciones a partir de una.

```
DATA programa;
INFILE 'c:\temp\clases.txt';
INPUT dia $10. clase $10. sec $ @;
OUTPUT;
INPUT clase $10. sec $ @;
OUTPUT;
INPUT clase $10. sec $ ;
OUTPUT;
PROC PRINT DATA=programa NOOBS;
TITLE 'Programa de clase';
BY dia;
ID dia;
RUN;
```